Proceedings of the Second International Conference on Electronics and Sustainable Communication Systems (ICESC-2021)
IEEE Xplore Part Number: CFP21V66-ART; ISBN: 978-1-6654-2867-5

2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC) | 978-1-6654-2867-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICESC51422.2021.9532917

Containerization: Cloud Computing based Inspiration Technology for Adoption through Docker and Kubernetes

Sanjay Hardikar
Scientist- F
National Informatics Centre
Ministry of Electronics & IT, GoI
Madhya Pradesh State Centre, Bhopal, MP
sjh@nic.in

Pradeep Ahirwar
Scientist- C
National Informatics Centre
Ministry of Electronics & IT, GoI
Madhya Pradesh State Centre, Bhopal, MP
ahirwar.pradeep@nic.in

Sameer Rajan
Scientist- C
National Informatics Centre
Ministry of Electronics & IT, GoI
Madhya Pradesh State Centre, Bhopal, MP
sameer.rajan@nic.in

***Abstract*-- The field of cloud computing has been evolving rapidly since inception. Cloud is the virtual pool of resources which can be served to the user through SaaS, PaaS and IaaS flavours. No cloud can be existed without virtualization. With Virtual Machine, the bare metal is virtualized to run multiple Operating System instances. These VMs serve to the users for performing the tasks. All are independent units and user has complete ownership and control to install the required softwares and use as per the wish. The VM solves many problems by optimizing the resources. The developers concern that the code is working fine on the development environment, but fail to work on testing or production environment due to the environment differences, if any.**

So, the Containerization comes into the picture to address such challenges. In this paper, various aspects of Containerization are explored and highlighted. The Container runtime environment- Docker and Container orchestration tool- Kubernetes are focused and deployed for exploring the possibilities of Containerization adoption, which automate the Container deployment, scaling and load balancing.

Keywords*-- *Cloud Computing, Containerization, Docker, Kubernetes, Microservices, Pod, Rancher.

I. INTRODUCTION

[1] Cloud Computing makes a virtual pool of resources which serves the user's requirement at reasonable cost. It manages the pool of resources automatically and dynamically and provides Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) through Public, Private and Hybrid Cloud models. It serves faster, simpler and cheaper services with high availability and scalability in elastic manner. In short, cloud serves 'Everything as a Service'.

[2] To harness the Cloud Computing benefits, Government of India already launched Meghraj Cloud. This not only makes certain the optimal infrastructure utilization but also expedite the deployment of e-governance applications.

[3] Cloud environment uses virtualization technology to migrate physical environment into virtual environment which reduces the overheads to maintain the hardware. Virtualization can be achieved by creating Virtual Machines (VMs). VM is the heavy weight resource and run top of the virtualization software and this virtualization software run on host operating system which slowdowns the performance. Each VMs created on top of the virtualization software have guest OS. User wants to run the application but as VM is equipped with OS and system's programs and files, these components consume space and computing power.

One of the common problems is faced by the developers and operational team that the code is working successfully on a machine but doesn't work on other one due to the differences in computing environment, if any.

As per current technology trend, the Microservices are gaining the popularity among the developers. Microservices address the issues associated with monolithic applications.

[4] Microservices are independent and self- sufficient components that isolate fine grained business capabilities. Microservices are based on Microservices Architecture which is an approach to design software applications as suites of independently deployable services. Many applications are easier to create and maintain if they are converted into smaller components which work in conjunction. Each component is developed independently, and the application is then just the integration of these individual components. The updations of application component are easy to handle. Also, the failure of one Microservice will not affect other Microservices of the system because of its loosely coupled nature. Microservice got a lot of popularity because of its advantages.

The Virtual Machine or Virtualization is not the suitable option to handle these challenges. To address such issues, the need of Containerization arises. Containerization facilitates to deploy various applications utilizing the same OS on a single VM/ Server. A Container uses operating system level virtualization for deploying applications instead of creating an entire VM.

[5] Containerization is a technology to virtualize the applications in a way that results the significant enhancement in cloud applications management.

[6] The Containers enveloped an application and the associated dependencies inside its own environment. It permits them to execute in isolated way while utilizing the

Authorized licensed use limited to: Ryerson University Library. Downloaded on August 11,2025 at 21:40:43 UTC from IEEE Xplore. Restrictions apply.

same resources including the Operating System. It enables a rapid, lightweight application deployment because the resources are utilized in feasible manner by not wasting on running separate operating system.

[7] The applications are enveloped with all the required dependencies into a standardized format called as a Container. These Containers carry on running in an isolated manner on top of the host OS. Now several Microservices can run in the same VM by running various Containers for each Microservice. In this way, it supports the Microservices architecture as well as resolves the issue of differences in computing environment on VM/ Servers. The application will always work same way without any difference that on which Platform/ OS (Data Center, Cloud, Windows and Linux Distros) it runs. Containerization is in vogue with the rise of Microservices and Docker.

There are various Containerization tools available to build, deploy and run the applications. Some Container runtime environments are Docker, Containerd, CRI-O etc.

[8] The Docker is the leading Container platform that packages code and dependencies together and ships them as a Container Image. Docker is an open source tool which also provides online repository of Docker Images called as Docker Hub. Docker Containers are simply runtime instances of Docker Images.

The complex requirements for a Microservice are written in easy to write DockerFile. The developer writes the codes that define the application requirement with its dependencies in a DockerFile. This DockerFile produces an Image. All the dependencies which are required for the application are present in the Image. This Image can be uploaded in Docker Hub. Docker hub is Git repository, which is distributed Version Control System, for Docker images. Anybody can pull the required Image and build a Container. There is no need to pre-allocate any RAM in the Containers.

The Containers could not communicate with each other so it is needed to deploy and manage them appropriately. They cannot be auto scaled to handle the workloads and distribution of traffic is also challenging with Containers.

So, to handle such issues, [9] Kubernetes comes into the picture. It is an open source Container orchestration engine which automates the Container deployment, Container scaling and Container load balancing. The basic deployment unit in Kubernetes is Pod. Pod is the basic unit in Kubernetesas Container is the basic unit in Docker. Each Pod contains mainly single Container.

II. MOTIVATION AND OBJECTIVE

Containerization can replace the tradition of virtual machines and monolithic approaches. The virtual machine is not optimal platform for resource utilization.

The VMs require separate OS and other resources to deploy and run the applications. The developers are also dependent on VM's environment to test, run and deploy the application and waste their time to sort out the issues of environment differences.

The microservices architecture enables the independent deployable services to take over and addresses the issues associated with the monolithic architecture. Microservices architecture makes the application loosely coupled by breaking a software application into independent deployable services.

To address the issues and to take the advantages of microservices, the Containerization plays a vital role. The applications are enveloped with all related dependencies into a format named as a Container. The developers take its advantages as the application is now tested, run and deployed without worrying about environment differences. Each service is deployed in separate Container to make it loosely coupled and independent deployable units.

There are many Container Runtimes but Docker is the most popular and open source tool, and as a Container Orchestration Platform Kubernetes is freely available and trending. Nowadays many IT giants adopting these latest trends and technologies, also many businesses are adopting this Containerization technology due to its easy deployment, scaling, and operations.

Here, the objective is to discuss the aspects of Containerization using Docker and Kubernetes and the focus is to deploy, run and test the Docker and Kubernetes in order to explore them and gaining the experience by deploying the applications..

III. DOCKER

Docker [10] works as Client- Server based model. The Docker client sends the request to Docker server/daemon. The Docker client and server both can be run on same or separate machine.

Docker makes it easier to create, deploy and run applications by using Containers. It simplifies the configuration, manage code pipeline, increase productivity, provide application isolation and mobility etc.

A. Docker Images

The Docker Image can be created using DockerFile. It contains a list of instructions. Simple Docker build command is sufficient to create an image file. All instructions written in the DockerFile are followed to build an Image.

B. DockerRepository

Docker images are placed in online repository of images. Images can be pushed and pulled through simple Docker push and Docker pull command respectively. Docker Hub is the repository where anybody can push or pull the images without building an Image afresh.

C. DockerContainer

A Docker Image is utilized to create a Docker Container. Containers contain all dependencies and instructions for an application to run in isolation. The Docker run command is available to execute the Docker Image to create a Container.

IV. KUBERNETES (K8S)

Authorized licensed use limited to: Ryerson University Library. Downloaded on August 11,2025 at 21:40:43 UTC from IEEE Xplore. Restrictions apply.

It is an open source project to automate deployment, scale, and to manage the Containerized applications [11]. Without the Container orchestration the human cost of running services was high, increased complexity of running something new in production, scaling was difficult, manual service setup and manual node crash fixing was required. Kubernetes (K8S) provides various significant features which allow running immutable infrastructure.

The key K8S features are as below-

- Horizontal scaling
- Automated rollouts and rollbacks
- Service discovery and load balancing
- Storage orchestration
- Secret and configuration management
- Self-healing
- Batch execution
- Automatic bin packing

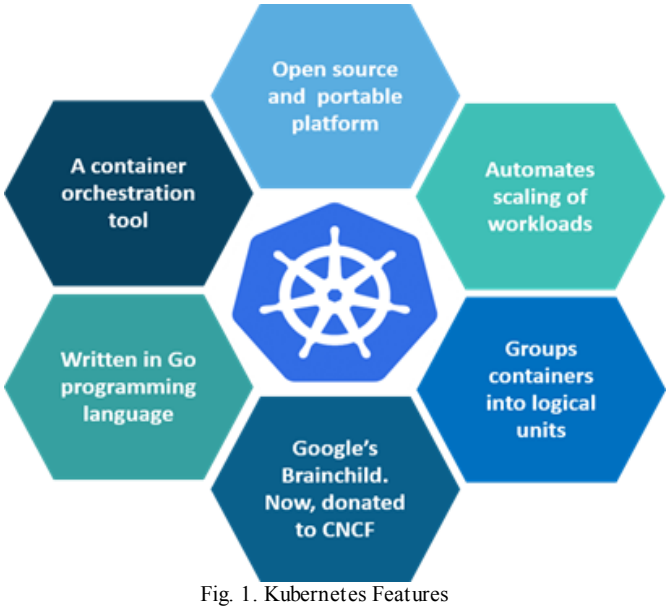

Fig. 1. Kubernetes Features

A. Pods

As the atomic unit of scheduling in Virtualization is VM and Container in Docker environment same way the atomic unit Pod is used in Kubernetes.

Single VM can run multiple Pods and each Pod can contain one or more Container.

Pod has unique IP address and each Container inside the Pod run on separate port numbers. There are two types of communication: intra Pod and inter Pod communication. The Container inside the Pod can communicate through the same network namespace which means all the Containers inside the same Pod share the same IP address but different port number while in the case of inter Pod communication, each Pod has separate IP address and this unique Pod IP establish the inter Pod communication.

When a Pod fails/ dies then Containers of that Pod are shifted to new Pod for uninterrupted execution. Pod is the basic scheduling unit and has unique IP address but Pods are ephemeral so if Pod fails, new Pod gets a fresh IP address so how to expose application to the outside world, how various components connect/ communicate and how to resolve Pod IP changes. To address these situations, Service takes the charge.

A service in Kubernetes is an abstraction that represents a logical set of pods and a policy through which they are accessed. The services correspond to a set of pods using labels and selectors. ClusterIP, NodePort and LoadBalancer are the main types of Services.

- ClusterIP exposes on an internal IP in the cluster. This is default Service and not accessible from outside the cluster.
- NodePort helps to expose the applications outside the world. It makes a Service accessible from outside the cluster.
- LoadBalancer provisions an external load balancer in current cloud and allocate a fixed, external IP to the Service.

B. Kubernetes Components

The cluster is composed of a master node, which helps in exposing the API, scheduling the deployment, and typically managing the cluster. Multiple worker nodes may be accountable for Container runtime (Docker, rkt etc.), plus an agent that talk to the master.

C. Master Node Components

- Kube-apiserver responsible for exposing the API.
- Etcd. Key value stores all the cluster data.
- Kube-scheduler schedules new Pods on worker nodes.
- Kube-controller-manager runs the controllers.
- Cloud-controller-manager talks to cloud providers.

D. Worker Node Components

Kubelet agent ensures that Containers in a Pod are running. Kube-proxy keeps network rules and performs forwarding. Container runtime runs the Containers.

E. Use Case of Kubernetes (Containerization)

[12] Secure, Scalable and Sugamya Website as a Service (S3WaaS) is a GIGW based website generating framework which is based on SaaS model hosted on National Cloud of National Informatics Centre (NIC). It is built on Software Defind Infrastructure for smooth provisioning hosting, compute, storage and networking.

It leverages technology for generating secure websites which are highly customizable and seamlessly deployed on a scalable and completely software defined infrastructure. All the district administration and many other governemnt bodies have launched the S3WaaS based websites, which are utilizing the benefits of cutting edge Containerization technologies like Docker, Kubernetes etc. and other latest trends.

Authorized licensed use limited to: Ryerson University Library. Downloaded on August 11,2025 at 21:40:43 UTC from IEEE Xplore. Restrictions apply.

The infrastructure procurement/ alocation and its management is not the matter of concern for user, the user focuses entirely on content quality. The website is developed on S3WaaS framework using CMS like WordPress provided through S3WaaS interface. User need to focus on website development instead of resource management. It is built on Containers using popular open source Container runtime- Docker and Container orchestration- Kubernetes tools.

Some important points related to the Infrastructure are mentioned as-

- Built on Containers using open source tools (Docker, Container, WordPress etc.)
- Agile and Scalable with Auto Healing and Auto Scaling.
- Seamless deployment, scaling and monitoring of each generated website.
- Static content published on an software defined Object Storage with replication.
- Automatic configuring of each website on software defined Load Balancer.
- Automatic updates for security patches.
- Dashboard for real time log monitoring and website analytics for insights into website usage.

V. RANCHER

Rancher [13] is a multi-cluster orchestration platform which sorts out the operational as well as security challenges to manage multiple Kubernetes clusters across any infrastructure, while making available DevOps teams with integrated tools for running Containerized workloads. It facilitates for delivering the Kubernetes-as-a-Service.

Rancher quickly deploys Kubernetes clusters anywhere on any provider and also unites these deployed clusters under centralized authentication and also the access control. Because it is agnostic about where the resources run, you can easily bring a cluster to a different provider and transfer resources between them. Instead of having multiple independent Kubernetes deployments, Rancher integrates them as a single, managed Kubernetes cloud.

Kubernetes is a powerful engine to orchestrate Containers. Rancher incorporates a full Kubernetes distribution, but adds value around Kubernetes in three crucial fields as Cluster Operations and Management, Intuitive Workload Management and Enterprise Support.

Following are some features of the Rancher [14].

- It supports the multi orchestration engines like Kubernetes, Cattle or Docker Swarm.
- It is useful to create a private SDN for each environment which enables secure communication.
- It distributes the traffic between Containers or services through Container load balancing.
- It supports orchestrating the persistent storage services for Docker.
- It is useful to implement distributed DNS based service discovery.
- It helps in monitoring the host resources and managing deployment of the Container.
- Rancher is designed for Multi-tenancy and user management support.
- It makes it easy to upgrade existing Container services, by allowing service cloning and redirection of service requests.
- Rancher supports Docker Machine, it monitors host resources and manages Container deployment which makes tasks more easier.

VI. STEPS DURING DEPLOYMENT SETUP

A. *Open Source Cutting Edge Technology Used*

Base OS- CentOS-8

Docker version: 19.03.13

Kubeadm /Kubectl Version: v1.19.2

Rancher v2.5.3

B. *Hands on to setupKubernetes Cluster with Docker Runtime Environment.*

The 04 nodes cluster with a master node has been set up as mentioned in Fig. 2. The master node works as manager node, while rest nodes work as worker node. The Workload is created on worker nodes only and the master node is responsible for managing the cluster. We follow the below mentioned steps to setup 04 Node Cluster

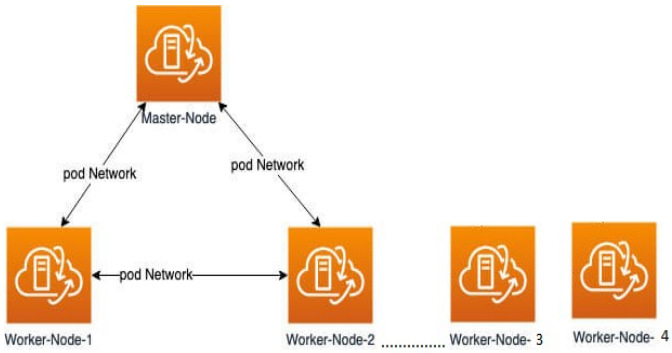

Fig. 2. 04 nodes cluster with a master

Step 1.
IP hostname mapping in /etc/hosts file on all (01+04) nodes.

Step 2.
systemctl stop/disable firewalld

Step 3.
Disable Swap to avoid some performance issue.

swapoff –a

Step 4.
Disable SELinux for subsequent reboots so that Kubernetes continues to run correctly

setenforce 0 (temporarily)
vi /etc/selinux/config (permanent)

Authorized licensed use limited to: Ryerson University Library. Downloaded on August 11,2025 at 21:40:43 UTC from IEEE Xplore. Restrictions apply.

SELINUX=Permissive

Step 5. Reboot the node machine

Step 6.
Docker installation

yum install docker –y

Step 7.
If RHEL/CentOS7 then following need to be done (Optional)

```
cat <<EOF > /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF

sysctl net.bridge.bridge-nf-call-iptables=1
sysctl net.ipv4.ip_forward=1
sysctl --system
echo "1" > /proc/sys/net/ipv4/ip_forward
systemctl daemon-reload
systemctl restart kubelet
```

Step 8.
Add Kubernetes Repo as the kubeadm, kubelet, kubectl packages not available with default OS Repository. So, it needs to create a repo file inside repos directory.

This repo file consists of location where related packages are available.

Run following on linux terminal.

```
cat <<EOF> /etc/yum.repos.d/Kubernetes.repo
[Kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/Kubernetes-e17-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpghttps://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
```

Step 9.
Kubernetes packages installation

yum install -y kubeadm

Step 10.
Now start/ enable and then see the status of Docker and kubelet service on nodes.

systemctl start/enable/status docker

systemctl start/enable/status kubelet

Note- Steps 1 to 10 are common for all worker nodes and master node.

Step 11.
Run following command to initialize the control plane node, only on Master Node

kubeadm init

Step 12.
Run following command only on Worker Nodes to add this worker node to the Kubernetes cluster

kubeadm join (generated token in step 11)

If the generated token is expired, obviously expired token will not work.

Step 13.
If token expired (24 hrs validity period) then run following command on Master Node to regenerate it and then do Step 12 on Worker Node.

kubeadm token create --print-join-command

Step 14.
Now Worker Node joined successfully, it can be verified from Master Node as kubectl get node. Fig. 3 highlights the basic details of all the worker nodes including master node.

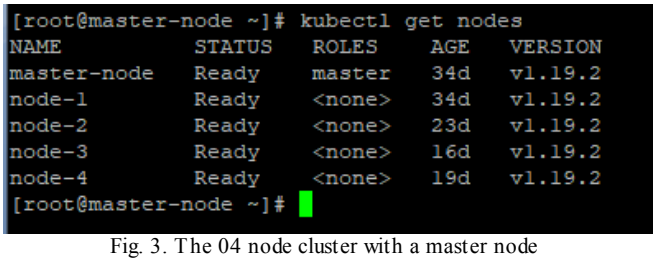

Fig. 3. The 04 node cluster with a master node

Later on, the same is verified through Kubernetes Dashboard utility. The dashboard screenshot of the created nodes is shown in the Fig. 4. The Kubernetes Dashboard is a web interface and the applications can also be deployed in the cluster, troubleshooting of the Containerized applications can be done and they can be easily managed. Dashboard provides an overview of the applications in the cluster. The Kubernetes resources can be easily created and modified with the help of the dashboard. It makes the task easier and quicker to manage the Kubernetes cluster.

The Kubernetes Dashboard may be deployed by running following command on Master Node:

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/aio/deploy/recommended.yaml

Then access it through browser as <nodeIP>:<nodePort>

Kubernetes Dashboard facilitates to create and deploy a Containerized application with a friendly wizard. The application details can be specified either manually or upload YAML or JSON file. [15] YAML (Ain't Markup Language) is a human-readable data-serialization language which is popularly used for configuration files and in applications where data is being stored or transmitted while JSON stands for JavaScript Object Notation. JSON is a lightweight format

Authorized licensed use limited to: Ryerson University Library. Downloaded on August 11,2025 at 21:40:43 UTC from IEEE Xplore. Restrictions apply.

and is used to store and transport data. The deployment is quicker through dashboard, user needs to just provide few details to launch the application and manage and troubleshoot it through the dashboard in a simpler way.

After that, Rancher unified cluster manager can also be deployed in order to manage the clusters. The complete workload can be managed for DevOps and to maintain corporate level security. By running the following command, multi- cluster orchestration platform Rancher can be launched to handle the ground to manage our cluster.

[16] docker run -d --restart=unless-stopped -p 80:80 -p 443:443 --privileged rancher/rancher:latest

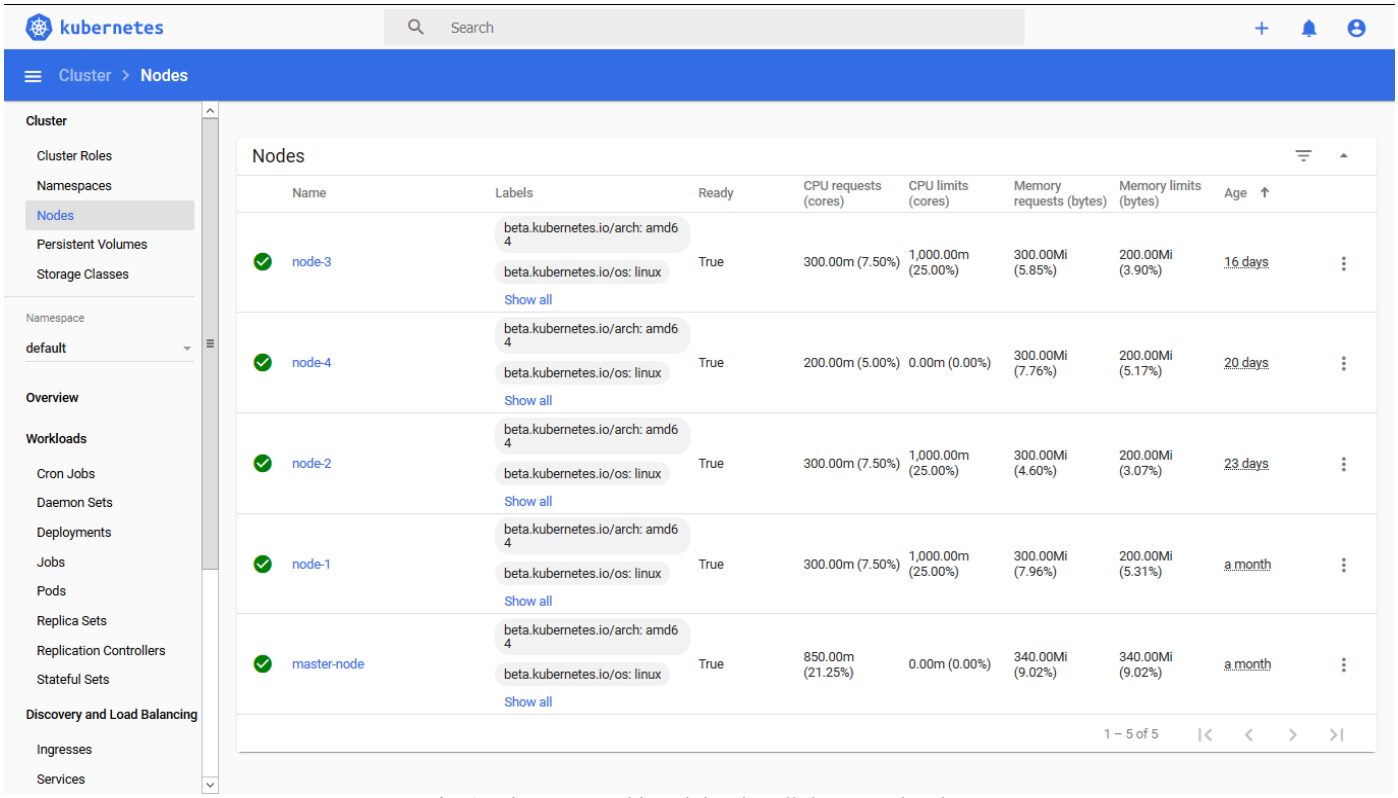

Fig. 4.Kubernetes Dashboard showing all the created nodes

Then the cluster can be added/ imported in Rancher orchestration manager. Fig. 5 shows the Rancher Dashboard of our added cluster which provides overview of the overall infrastructure and workloads etc. The figure shows the created resources like Namespaces, Deployments, Services, DaemonSets, Ingresses, Jobs and StatefulSet etc. of our cluster.

Authorized licensed use limited to: Ryerson University Library. Downloaded on August 11,2025 at 21:40:43 UTC from IEEE Xplore. Restrictions apply.

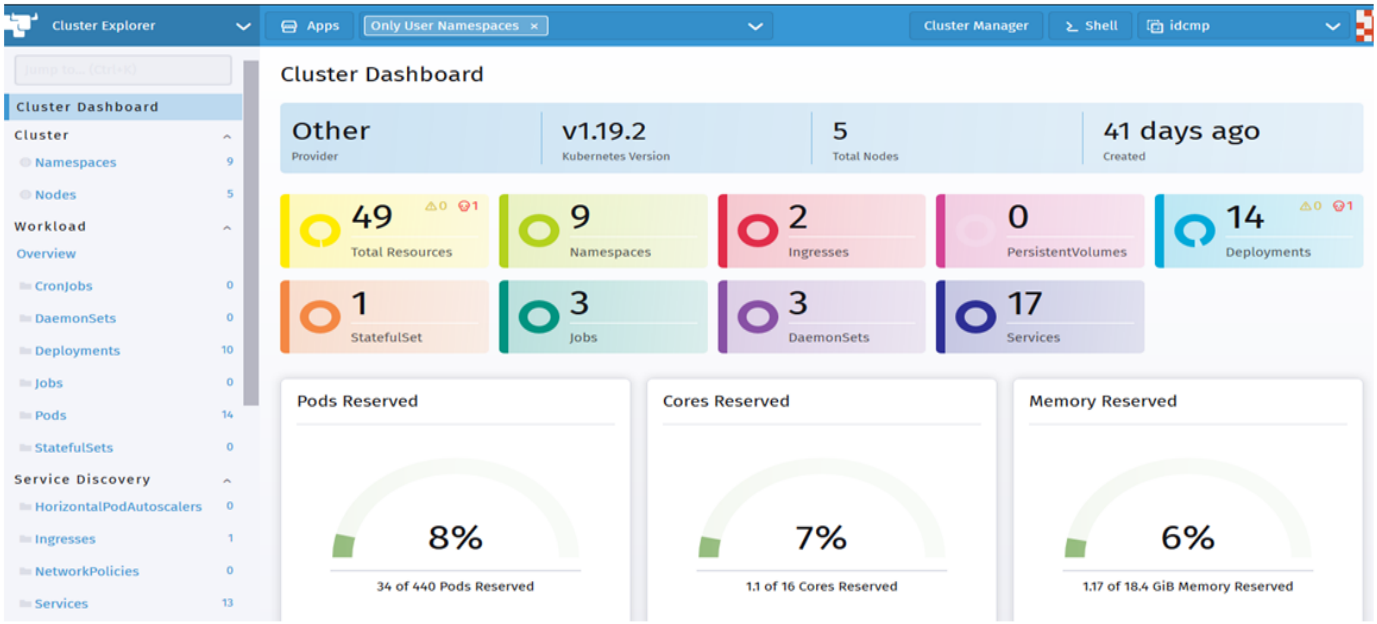

Fig. 5. Rancher Dashboard of the Cluster

C. Hands on to deploy NGINX on Kubernetes Cluster

Step 1.

A NGINX deployment is created with two replicas/Pods

kubectl create deployment testwebapp --image=nginx --replicas=2 --port=80

Step 2.

Check the deployment status

kubectl get deploy -o wide

Step 3.

Check the status of Pods

kubectl get Pod -o wide

Step 4.

Ensure the Pods are accessible from other node

curl <Pod IP address>:80

Step 5.

Create a Service to expose the port to outside the cluster

kubectl expose Pod testwebapp --port=8080 --target-port=80 --type=LoadBalancer

Step 6.

Check the Service status

kubectl get svc

Step 7.

It can also be checked by curl <nodeIP>:<nodePort>

VI. CONCLUSION

Virtualization technology is the fundamental technology for cloud environment. The limitations and problems associated with VMs and popularity of the Microservices give birth to utilize the leading- edge Container technology. Various Containerization tools are available to address the challenges towards the evolution in cloud environment. Docker is the leading Container runtime and Kubernetes is the Container orchestration tool, which are popular among the DevOps (Development and Operation) teams for adopting CI/ CD (Continuous Integration and Continuous Deployment) in Software Development Life Cycle (SDLC). Kubernetes A lot of companies and government organizations are utilizing the benefits of Containerization by implementing a DevOps mindset to their workflow. Pods work like the VMs to deploy the applications but in dynamic way without pre-allocating the resources or installing the supporting softwares to run the application.

So the applications are not dependent on computing environment. Such technologies are evolving day by day to ease the resource allocation and usage. This evolution is unstoppable and refinement is happening for transformation of the infrastructure allocation and utilization.

Today, it is mandatory to adopt these technologies of inspiration to serve and use the ICT resources in effective and efficient manner by reducing the human intervention, for moving towards the Software Defined Infrastructure.

REFERENCES

[1] Sameer Rajan and Apurva Jairath, "Cloud Computing: The Fifth generation of Computing, International Conference on Communication Systems and Network Technologies", IEEE, 2011.

[2] Dr. Saurabh Gupta and Sameer Rajan et al., "Cloud Based Integrated Bhulekh Model Transforming Revenue Administration and Digitally Empowering Society", EPH- International Journal of Science and Engineering, Vol. 2 (2), Dec- 2016.

[3] Srinath Reddy Meadusani, "Virtualization Using Docker Containers: For Reproducible Environments and Containerized Applications", Department of Information Systems, St. Cloud State University, May 2018.

[4] Markos Viggiato et al., "Microservices in Practice: A Survey Study" Dept. of Computer Science, UFMG and UFLA, MG Brazil.

[5] Claus Pahl et al., "Cloud Container Technologies: a State-of-the-Art Review", funded by University of Pisa and Bozen-Bolzano, May 2017.

Authorized licensed use limited to: Ryerson University Library. Downloaded on August 11,2025 at 21:40:43 UTC from IEEE Xplore. Restrictions apply.

[6] Containerization, "Virtualization vs. Containerization", available at https://www.liquidweb.com/kb/virtualization-vs-Containerization/ (Accessed on 05 Jan 2020).
[7] Babak Bashari Rad et al., "An Introduction to Docker and Analysis of its Performance", IJCSNS International Journal of Computer Science and Network Security, Vol. 17 (3), Mar-2017.
[8] Docker, "Container Runtime Engine", available at https://www.Docker.com/ (Accessed on 07-Feb-2020).
[9] Kubernetes, "Kubernetes- Container Orchestration Tool", available at https://Kubernetes.io/ (Accessed on 07-Mar-2020).
[10] Docker Documentation, "Docker Docs", available at https://docs.Docker.com/ (Accessed on 01-Apr-2021).
[11] What is Kubernetes, "Kubernetes Features", available at https://blog.csdn.net/u012679583/article/details/112967484 (Accessed on 06-04-2021)
[12] S3WaaS, "Secure, Scalable and Sugamya Website as a Service", available at https://cdn.s3waas.gov.in/master/uploads/2018/08/2018080176.pdf (Accessed on 07-Apr-2021)
[13] Rancher, "Rancher Innovate Anywhere", available at https://rancher com/ (Accessed on 08- Apr-2021).
[14] Rancher, "Key Features", https://rancher.com/docs/rancher/v1.0/en/ (Accessed on 08-Apr-2021)
[15] YAML, "YAML Ain't Markup Language", available at https://en.wikipedia.org/wiki/YAML#:~:text=YAML%20(a%20recursive%20acronym%20for,is%20being%20stored%20or%20transmitted (Accessed on 10-Apr-2021)
[16] Rancher Setup, "Rancher Install and Run", available at https://rancher.com/quick-start/ (Accessed on 11-Apr-2021)

Authorized licensed use limited to: Ryerson University Library. Downloaded on August 11,2025 at 21:40:43 UTC from IEEE Xplore. Restrictions apply.