A Survey on Kubernetes Architecture and Its Significance

Neha Sahu and Subhash Chandra Thakre

January 30, 2022

A survey on Kubernetes architecture and its significance

Neha Sahu[1] and Subhash Chandra Thakre[2]

[1] Lecturer (Computer Science), KGP Raigarh(C.G.)
neha.s1410@gmail.com
[2] Subhash Chandra Thakre, National Informatics Centre,
subhash.thakre@nic.in

Abstract. Orchestration and automation have become an integral part of container deployment and management. Kubernetes (also known as K8s) is a production-grade container orchestration software. It is an open source cluster management system. Kubernetes now orchestrates half of the container environments, and more organizations are migrating to managed services like Google Kubernetes Engine (GKE), Azure Kubernets Service(AKS) and Amazon Elastic Kubernetes Services(EKS). It rapidly going to become the new standard for maintaining and deploying software in the cloud. This paper discusses a detailed architecture of kubernetes, nodes, some of its basic concepts, services, advantages and limitations.

Keywords: Kubernet, pod, Docker, node, container, cluster.

1 Introduction

Kubernetes has become the fectual standard for container orchestration. Today almost half of running containers applications are using Kubernets, whether in self-managed clusters, or through a cloud provider service like Google Kubernetes Engine (GKE), Azure Kubernets Service(AKS) and Amazon Elastic Kubernetes Services(EKS). Kubernetes adoption has quite doubled since 2017 and continues to grow steadily, without any sign of slowing down.

1.1 Why kubernetes

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)[12]. VM is the heavy weight resource and run top of the virtualization software and this software run on host operating system which declines the performance.

Containerization[11] 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. The Containers[11] enveloped an application and the associated dependencies inside its own environment. It permits them to execute in isolated way while utilizing the same resources including the Operating System.

CONTAINER ORCHESTRATION [1]
When applications are developed in a way in which they form a network of modular services and are independently deployable and where each service operates on a different process and communicates over a lightweight process to meet an objective are termed as micro services and also called service oriented architecture (SOA). This type of architecture is now universally adopted in the industry with the help of containers. Containers are lightweight Virtualization technology firstly available in Linux based systems. When a container based system scales up, some major tasks which arise are:

Scheduling of containers
Management of life cycle
Load Balancing[1]
Service Discovery

The Containers could not communicate with each other, deployment and managing these lots of containers across multiple environments using scripts and self-made tools can be really complex. They cannot be auto scaled to handle the workloads and distribution of traffic is also challenging with them.
So, to handle such issues, Kubernetes comes into the picture. Kubernetes is powerful open-source orchestration software developed by google in 2014 for automating deployment, scaling and managing containerized workloads. It provides an API to control how and where containers will run. It permits us to run containers and workloads and helps to solve some of the working problems when moving to scale multiple containers, deployed across multiple servers. Kubernetes provide interfaces and composable platform that help us to define and manage the application with high degrees of reliability, power and flexibility.

2 Hardware components

2.1 Node

A node[4] is the smallest unit of computing hardware in Kubernetes. It is a representation of a single machine in the cluster. Each node can be visualized as a set of CPU and RAM resources. It may be either a physical machine in a datacenter, or virtual machine hosted on a cloud. Examples of nodes can be a computer system, mobile phones, smartwatches, etc.

2.2 Cluster

The cluster can be specified as a group of individual nodes which work together. In Kubernetes cluster[6], all the nodes share their resources to form a powerful machine. When programs are deployed onto the cluster, it smartly handles distributing work to the individual nodes. If any nodes are added or deleted, the cluster will move around work as per need. It doesn't matter to the program, or the programmer, which individual machines(node) are actually running the code.

2.3 Persistent Volume

It is a basic building block of the kubernetes[4] storage which is used to store information persistently. It provides a file system that can be lived within the cluster without being attached to any individual node. Regular small volumes are mounted on individual nodes that are used to share information among nodes. These regular volumes are deleted when the pods are removed, but a persistent volume can remain alive for as long as necessary.

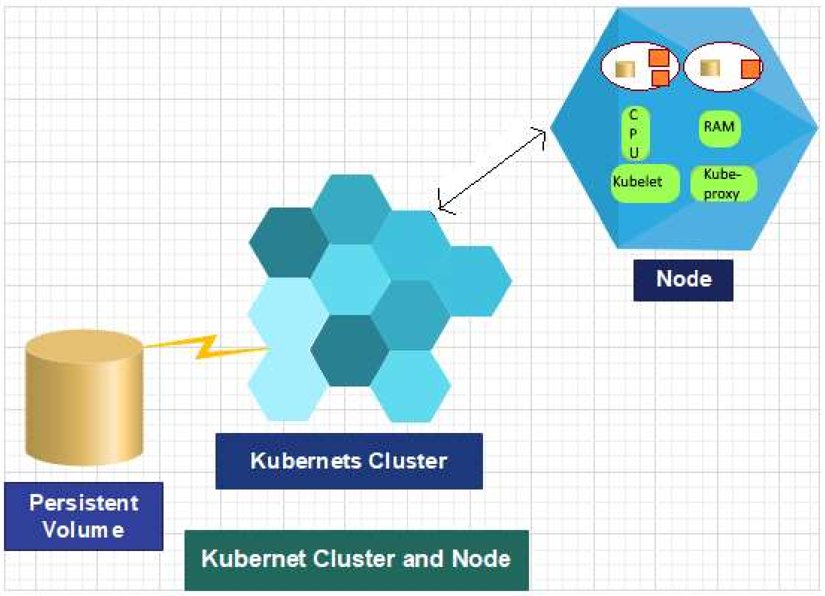

Fig.1: Kubernetes cluster and node

3 Kubernetes architecture

Kubernetes cluster[4] architecture consists of at least one master node and multiple worker nodes. There may be more than one master node for high availability and fault tolerance. Kubernetes cluster can support 5000 worker nodes per cluster.

3.1 Worker Node components

There are basically five major components of worker nodes[2] concerning kubernetes. They are responsible for maintaining and running pods.

Pod. Pods[4] are the smallest deployable units in Kubernetes. A Pod represents a single instance of a running process in the cluster. Pods contain one or more containers. When a Pod runs multiple containers, all the containers are treated and managed as a single entity and share the Pod's resources. Pods also contain shared storage and networking resources for their containers:

Storage. Pods can specify shared storage volumes that can be shared among all the containers of a pod.

Network. Pods are automatically assigned unique IP addresses. All the containers of one pod share the same network namespace, including IP address and network ports. Containers in a Pod can communicate with each other on localhost.

A Pod [7] is meant to run a single instance of the application on the cluster. However, it is not suggested to create individual Pods directly. Instead, we normally create a set of identical Pods, which are called replicas, to run the application. Such a set of replicated Pods are created, deployed and managed by a controller. Controllers are used to manage the lifecycle of their Pods and can also perform horizontal scaling, deleting and creating new Pods as necessary. Pods run on nodes in the cluster. Once created, a Pod remains on its node until its procedure is completed, the Pod is deleted or removed from the node due to lack of resources, or the node failures. If a node fails, all the pods on that node are automatically scheduled for deletion.

Container. A software container[4] is a method of packing up an application or service and its associated dependencies so the application runs rapidly and reliably from one computing environment to another. A Container image is a standalone , lightweight, executable package of software which contains everything required to run an application code, runtime, system tools, system libraries and settings. The Container image is like class and containers are its object, so we can create many containers with one image.

Container runtime. It is a software that run containers and manages images on a node. Kubernetes supports several container runtimes: Docker, containerd, Rocket CRI-O, and any implementation of the Kubernetes CRI (Container Runtime Interface).

Kubelet. It is the primary node agent which works on every single worker node. It is responsible for creating, deploying and managing pods and their containers. It deals with pods specification which is defined in YAML or JSON format. With the help of kubelet master node one can monitor worker nodes' pod specification and check whether the pods are running healthy or not. It is a service responsible for communication with the control panel service(API server). It interacts with ETCD and Master node to get commands and work efficiently.

Kube-proxy. It is a key component of kubernetes architecture and runs on every single node. Its task is to balance load and manage the traffic that is predestined for services to the correct backend pods. Which gives services to the outside world. It is truely the core networking component.

3.2 Master node components

The Master node[2] is responsible to manage the whole cluster. There may be more than one master node. It monitors health checks of nodes. It shows information about members of the cluster and its configuration inside master node. So when a worker node fails, it transfers workload from failed node to other healthy worker node. Kubernetes master node is responsible for scheduling, managing, provisioning, and exposing API to the client. It has four major components:

API server. Kube-apiserver is a kind of gatekeeper for the entire cluster, which validates and constructs the API objects such as pods, services, deployments, replication controllers and others. The Kubernetes API works as the front end of the Kubernetes control plane and designs the way how users interact with their cluster. The API (application programming interface) server determines if a request is valid and then processes it.

In essence, the API server is the interface which is used to create, ,manage, update, delete or display any kubernetes objects. It provides a way through which the users, external components, and parts of our cluster all communicate with each other.

Kube-Scheduler. it is a monolithic component of kubernetes decoupled from the API server. It is used for physical scheduling [5] of pods across multiple nodes. For every newly generated pod or other unscheduled pods, it is the task of kube-scheduler to choose an optimal node for them to deploy and run on. However, each container in pods has separate requirements for resources and every single pod also has separate requirements. Therefore, existing nodes need to be filtered according to the particular scheduling requirements.

In a cluster, Nodes that fulfill the scheduling requirements are called feasible nodes. If none of the nodes are suitable or meeting the requirements, the Pod remains unscheduled until the scheduler finds the new feasible node. When the scheduler finds a number of feasible Nodes for a Pod then it runs a set of functions to score the feasible Nodes and picks a Node with the highest score among the all to run the Pod. The scheduler then notifies the API server about this decision made in a process called binding.

Control manager. There are four controllers behind control manager, node controller, replication controller, point controller, service account and token controller. These all are responsible for the overall health management of the entire cluster. It ensures all nodes and correct number of pods are up and running.

Etcd. It is a distributed key value database. Etcd is like a cluster brain. It is a central database used to store the current cluster state at any point of time. Any module of kubernetes can query etcd about the state of the cluster. It is the single state of cluster. It is the single source for all kubernetes components.

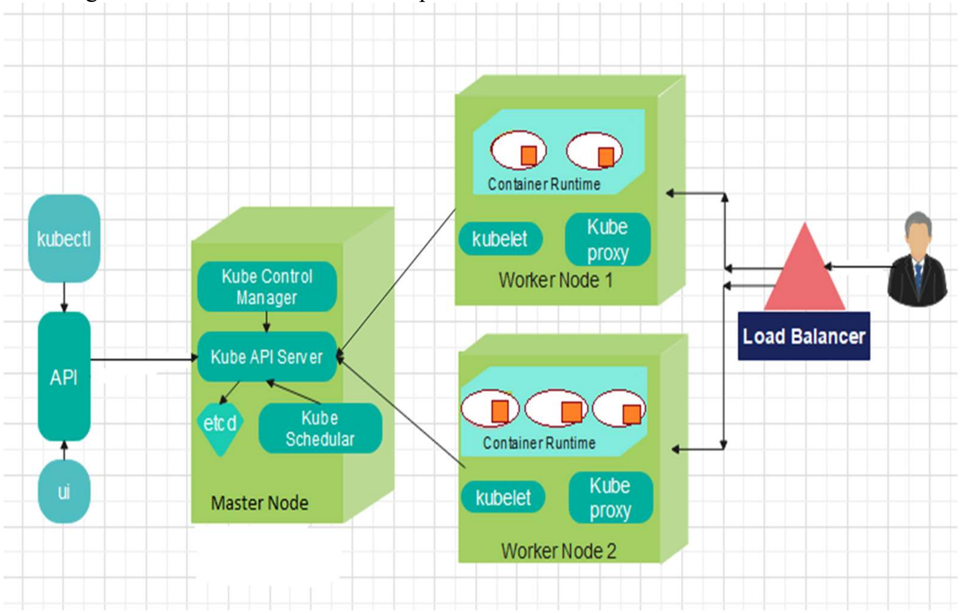

Fig.2: Kubernetes Architecture

4 Services provided by Kubernetes

As applications get bigger multiple containers deployed across multiple servers, managing them will become more complex. To cope up this complexity, Kubernetes provides an open source API that controls where and how those containers[3] will run. Kubernetes orchestrates clusters of virtual machines and schedules[5] containers to run on those machines based on their available compute resources and the requirements of each container. Containers are grouped into pods, the basic operational unit for Kubernetes and these pods scale to the desired state. The current status of each pod is always being compared with the target status and being modified when necessary. These processes are performed automatically.

Kubernetes automatically manages service discovery, security, self-healing, incorporates load balancing, traces resource allocation and scales up and down based on compute utilization. And, it checks the health of individual pods, containers and resources and enables apps to self-heal by automatically restarting, stopping on failures or replicating containers when needed.

4.1 Significance of Kubernetes

While the containers guarantees to code once and run anywhere, Kubernetes gives the potential to manage and orchestrate all the container resources from a single control plane. It helps in networking, load-balancing, auto-healing, security, and auto-scaling across all Kubernetes nodes which run the containers[3]. Kubernetes also has a built-in isolation mechanism like namespaces which allows us to group container resources by access permission, staging environments, and more. These constructs make easy for IT to allow developers with self-service resource access and developers to cooperate on even the most complex microservices architecture without mocking up the entire application. We can combine DevOps practices with containers and Kubernetes, which further enables a guideline for microservices architecture that encourages fast delivery and scalable orchestration of cloud-native applications.
It makes infrastructure more robust and applications highly available. The application will persist online, even if some of the nodes go offline. If our application starts to get a lot more load and needs to scale out to be able to provide a better user experience, it is simple to scale up more containers or add more nodes to the Kubernetes cluster.
Kubernetes and Docker[4] work together. Docker provides the ability for packaging and running containerized applications. Using Docker, we can build and run containers and store and share container images. One can easily run a Docker[4] image build on a Kubernetes cluster, but Kubernetes itself is not a complete solution. To optimize Kubernetes services in production, execute additional tools, technologies and services to manage security, governance, identity, and ingress along with continuous integration/continuous deployment (CI/CD) workflows.

5 Advantages of Kubernetes

Some advantages of Kubernetes are as follows:

5.1 Portability and flexibility[10]

Kubernetes can work with virtually any type of container runtime. In addition, it can work with any type of underlying infrastructure whether it is an on-premises server, a public cloud, or a private cloud.

In these respects, Kubernetes is highly portable and flexible, because it can be used on a variety of environment configurations and different infrastructure. Most other orchestration tools lack this portability; they are tied to a particular environment configuration or infrastructures.

5.2 Multi-cloud capability

In terms of capability, Kubernetes can host workloads running on a single cloud as well as workloads that are spread across multiple clouds. In addition to this, Kubernetes can easily scale from one cloud to another.

5.3 Open source

Kubernetes is a fully open source, community-led project overseen by the CNCF(Cloud Native Computing Foundation). It has a number of major corporate sponsors, but no one company owns it. In 2019, Weaveworks was one of the top eight Kubernetes contributors in the CNCF's Kubernetes Project Journey report [10].

5.4 Automates various manual processes

For instance, Kubernetes places containers based on the resource requirements. It will control which pod will host the container, how it will be launched etc.

5.5 Interacts with several groups of containers

Kubernetes is able to manage more than one clusters at the same time.

5.6 Provides additional services

As well as the management of containers, Kubernetes also offers security, load balancing, networking and storage services.

5.7 Self-monitoring and self-healing

Kubernetes checks timely the health of nodes and containers and replaces or reschedules them when node die, stop the containers that don't respond.

5.8 Horizontal scaling

Kubernetes allows us to scale the application not only vertically but also horizontally, easily with a simple command, a UI or automatically.

5.9 Storage orchestration

Kubernetes automatically mounts and adds storage system of our choice to run apps.

5.10 Automates rollouts and rollbacks

Kubernetes can roll out changes to the application or its configuration. If after a performing a change to the application something goes wrong, we can roll back with the help of Kubernetes.

5.11 Container balancing

Kubernetes always knows how to compute the optimal location for the containers and how to place them. It helps to manage traffic management services efficiently.

5.12 Run everywhere

Kubernetes is an open source container orchestration tool and gives us the freedom to take advantage of public cloud, on-premises, or hybrid infrastructure, allowing us to move workloads to anywhere

6 Limitations[8]

6.1 Kubernetes can be an overkill for simple applications

Kubernetes is a complex but powerful technology that allows us to run software in a cloud environment at a large scale pretty efficiently. However, if we do not aim to develop anything complex for a large or distributed or with high computing resource needs, this is not much beneficial for us from the technical power of k8s.

6.2 Kubernetes is very complex and can reduce productivity

Kubernetes is especially known for its complexity. Specially for developers who are not familiar with infrastructure technologies of Kubernetes, it can be very hard to adapt the Kubernetes development workflow.

6.3 The transition to Kubernetes can be cumbersome

Since most companies cannot start on a green field, the existing software should be adapted to run smoothly with Kubernetes or we must perform some changes in the newly built application that will run on Kubernetes. It is hard to estimate how much time and effort this requires as this depends heavily on the application software.

6.4 Kubernetes can be more expensive than its alternatives

It can be more expensive than its alternatives. This is because all of the previously mentioned disadvantages cost the time of expert engineers. Besides this indirect cost, sometimes the infrastructure cost of using Kubernetes is higher than for alternatives, mainly for small applications as Kubernetes itself has some advanced computing needs.

7 Conclusion

The main purpose of writing this paper was to give users a brief overview architecture of Kubernetes and its important components. It is an exciting project that permits users to run highly available, scalable, containerized workloads on an extremely abstracted platform. While architecture of Kubernetes first seems daunting, their flexibility, power and robust feature set are unmatched in the open source world. By understanding Kubernetes basic building blocks one can design the system that fully leverage the capabilities of the platform. Kubernetes provide bare minimum services to manage a few thousand containers across multiple data handling centers, but as this technology comes in demand, the requirements start to vary. The continuous monitoring of containers is a challenge, security is one of the major concerns and so on. In future, we will try to look into the microservices deployment and how Kubernetes behave on the failure of the worker nodes or master nodes. We believe, with time this technology will get matured and will be deployed more widely.

References

1. Hritwik Bairagi, Uday Chourasiya, Sanjay Silakari, Priyanka Dixit, Smita Sharma, A Survey On Efficient Container Orchestration Tools And Techniques In Cloud Environment, INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 9, ISSUE 01 (2020).

2. Nikhil Marathe, Ankita Gandhi, Jaimeel M Shah, Docker Swarm and Kubernetes in Cloud Computing Environment, Proceedings of the Third International Conference on Trends in Electronics and Informatics (ICOEI 2019).
3. Vivek Sharma, Harsh Kumar Saxena, Akhilesh Kumar Singh, Docker for Multi-containers Web Application, Second International Conference on Innovative Mechanisms for Industry Applications (ICIMIA 2020)
4. Mohammad Ahmadi, "An Introduction to Docker and Analysis of its Performance", IJCSNS International Journal of Computer Science and Network Security,VOL.17 No.3, March 2017.
5. Oana-Mihaela Ungureanu, Călin Vlădeanu, Robert Kooij, Kubernetes cluster optimization using hybrid shared-state scheduling framework, ICFNDS '19, July 1–2, 2019
6. https://www.weave.works/blog/kubernetes-cluster, last accessed on 2022/01/03.
7. https://cloud.google.com/kubernetes-engine/docs/concepts/pod, last accessed on 2022/01/02
8. https://devspace.cloud/blog/2019/10/31/advantages-and-disadvantages-of-kubernetes, last accessed on 2021/12/25
9. https://www.hitechnectar.com/blogs/pros-cons-kubernetes/, last accessed on 2021/11/25
10.https://www.weave.works/blog/6-business-benefits-of-kubernetes, last accessed on 2021/12/29
11. Containerization, "Virtualization vs. Containerization", https://www.liquidweb.com/kb/virtualization-vs-Containerization/, last accessed on 2021/12/27
12 . https://www.vmware.com/topics/glossary/content/virtual-machine.html, last accessed on 2022/01/13