**INTRODUCTION:**

Kubernetes is an open-source, container management solution originally announced by Google in 2014. After its initial release in July 2015, Google donated Kubernetes to the Cloud Native Computing Foundation. Since then, several stable versions have been released under Apache License.

For a developer, Kubernetes provides a manageable execution environment for deploying, running, managing, and orchestrating containers across clusters or clusters of hosts. For devops and administrators, Kubernetes provides a complete set of building blocks that allow the automation of many operations for managing development, test, and production environments. Container orchestration enables coordinating containers in clusters consisting of multiple nodes when complex containerized applications are deployed. This is relevant not only for the initial deployment, but also for managing multiple containers as a single entity for the purposes of scaling, availability, and so on.

Being infrastructure agnostic, Kubernetes clusters can be installed on a variety of public and private clouds (AWS, Google Cloud, Azure, OpenStack) and on bare metal servers. Additionally, Google Container Engine can provide a deployed Kubernetes cluster. This makes Kubernetes similar to Linux kernel, which provides consistency across different hardware platforms, or Java, which runs on almost any operating system.

**KUBERNETES — HIGH LEVEL ARCHITECTURE**

**NODE :**

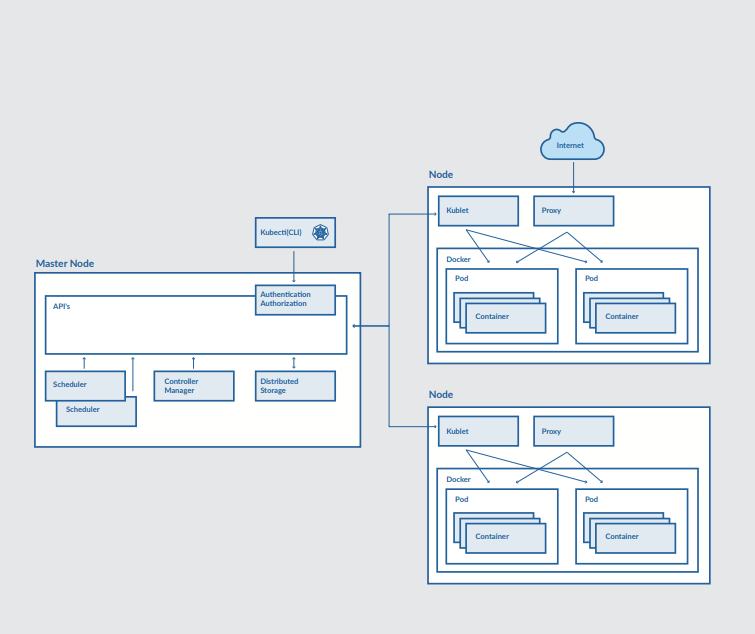
A Kubernetes cluster consists of one or more nodes managed by Kubernetes. The nodes are bare-metal servers, on-premises VMs, or VMs on a cloud provider. Every node contains a container runtime (for example, Docker Engine), kubelet (responsible for starting, stopping, and managing individual containers by requests from the Kubernetes control plane), and kube-proxy (responsible for networking and load balancing).

**MASTER NODE** :

A Kubernetes cluster also contains one or more master nodes that run the Kubernetes control plane. The control plane consists of different processes, such as an API server (provides JSON over HTTP API), scheduler (selects nodes to run containers), controller manager (runs controllers, see below), and etcd (a globally available configuration store).

**DASHBOARD AND CLI:**

A Kubernetes cluster can be managed via the Kubernetes Dashboard, a web UI running on the master node. The cluster can also be managed via the command line tool kubectl, which can be installed on any machine able to access the API server, running on the master node. This tool can be used to manage several Kubernetes clusters by specifying a context defined in a configuration file



**KUBERNETES BUILDING BLOCKS**

Kubernetes provides basic mechanisms for the deployment, maintenance, and scaling of containerized applications. It uses declarative primitives, or building blocks, to maintain the state requested by the user, implementing the transition from the current observable state to the requested state.

**POD:**

A pod is the smallest deployable unit that can be managed by Kubernetes. A pod is a logical group of one or more containers that share the same IP address and port space. The main purpose of a pod is to support co-located processes, such as an application server and its local cache. Containers within a pod can find each other via localhost, and can also communicate with each other using standard inter-process communications like SystemV semaphores or POSIX shared memory. In other words, a pod represents a “logical host”. Pods are not durable; they will not survive scheduling failures or node failures. If a node where the pod is running dies, the pod is deleted. It can then be replaced by an identical pod, with even the same name, but with a new unique identifier (UID)

**LABEL:**

A label is a key/value pair that is attached to Kubernetes resource, for example, a pod. Labels can be attached to resources at creation time, as well as added and modified at any later time

**SELECTOR:**

A label selector can be used to organize Kubernetes resources that have labels. An equality-based selector defines a condition for selecting resources that have the specified label value. A set-based selector defines a condition for selecting resources that have a label value within the specified set of values

**CONTROLLER:**

A controller manages a set of pods and ensures that the cluster is in the specified state. Unlike manually created pods, the pods maintained by a replication controller are automatically replaced if they fail, get deleted, or are terminated. There are several controller types, such as replication controllers or deployment controllers

**REPLICATION CONTROLLER:**

A replication controller is responsible for running the specified number of pod copies (replicas) across the cluster

**DEPLOYMENT CONTROLLER:**

A deployment defines a desired state for logical group of pods and replica sets. It creates new resources or replaces the existing resources, if necessary. A deployment can be updated, rolled out, or rolled back. A practical use case for a deployment is to bring up a replica set and pods, then update the deployment to re-create the pods (for example, to use a new image). Later, the deployment can be rolled back to an earlier revision if the current deployment is not stable

**REPLICA SET:**

A replica set is the next-generation replication controller. A replication controller supports only equality-based selectors, while a replica set supports set-based selectors

**SERVICE:**

A service uses a selector to define a logical group of pods and defines a policy to access such logical groups. Because pods are not durable, the actual pods that are running may change. A client that uses one or more containers within a pod should not need to be aware of which specific pod it works with, especially if there are several pods (replicas). There are several types of services in Kubernetes, including ClusterIP, NodePort, LoadBalancer. A ClusterIP service exposes pods to connections from inside the cluster. A NodePort service exposes pods to external traffic by forwarding traffic from a port on each node of the cluster to the container port. A LoadBalancer service also exposes pods to external traffic, as NodePort service does, however it also provides a load balancer

**STORAGE BUILDING BLOCKS**

**VOLUME:**

A container file system is ephemeral: if a container crashes, the changes to its file system are lost. A volume is defined at the pod level, and is used to preserve data across container crashes. A volume can be also used to share data between containers in a pod. A volume has the same lifecycle as the the pod that encloses it— when a pod is deleted, the volume is deleted as well. Kubernetes supports different volume types, which are implemented as plugins.

**PERSISTENT VOLUME:**

A persistent volume claim defines a specific amount of storage requested and specific access modes. Kubernetes finds a matching persistent volume and binds it with the persistent volume claim. If a matching volume does not exist, a persistent volume claim will remain unbound indefinitely. It will be bound as soon as a matching volume become available. A persistent volume represents a real networked storage unit in a cluster that has been provisioned by an administrator. Persistent storage has a lifecycle independent of any individual pod. It supports different access modes, such as mounting as read-write by a single node, mounting as read-only by many nodes, and mounting as read-write by many nodes. Kubernetes supports different persistent volume types, which are implemented as plugins. Examples of persistent volume types include AWS EBS, vSphere volume, Azure File, GCE Persistent Disk, CephFS, Ceph RBD, GlusterFS, iSCSI, NFS, and Host Path.

**PERSISTENT VOLUME CLAIM:**

A persistent volume claim defines a specific amount of storage requested and specific access modes. Kubernetes finds a matching persistent volume and binds it with the persistent volume claim. If a matching volume does not exist, a persistent volume claim will remain unbound indefinitely. It will be bound as soon as a matching volume become available

**CHOOSING THE RIGHT BLOCK FOR THE JOB:**

Designed as a simple building block; a replication controller’s only responsibility is to maintain the specified number of replicas. A replication controller counts only live pods;terminated pods are excluded. Other Kubernetes building blocks should be used together with replication controllers for more advanced tasks. For example, an autoscaler can monitor application specific metrics and dynamically change the number of replicas in the existing replication controller. In addition, a replication controller does not support scheduling policies, meaning you cannot provide rules for choosing cluster nodes to run pods from the managed set.

A replica set is another Kubernetes building block. The major difference between it and a replication controller is that replication controllers do not support selectors with set-based requirements, while replica sets support such A Kubernetes secret allows users to pass sensitive information, such as passwords, authentication tokens, SSH keys, and database credentials, to containers. A secret can then be referenced when declaring a container definition, and read from within containers as environment variables or from a local disk. A Kubernetes config map allows users to externalize application configuration parameters from a container image and define application configuration details, such as key/value pairs, directory content, or file content. Config map values can be consumed by applications through environment variables, local disks, or command line arguments. selectors. From this perspective, a replica set is just a more advanced version of a replication controller.

Using only pods and replication controllers to deploy an application is, at least in part, an imperative form of managing software, because it usually requires manual steps. A Kubernetes deployment is an alternative that enables completely declarative application deployment

**SECRET:**

A Kubernetes secret allows users to pass sensitive information, such as passwords, authentication tokens, SSH keys, and database credentials, to containers. A secret can then be referenced when declaring a container definition, and read from within containers as environment variables or from a local disk

**CONFIG MAP:**

A Kubernetes secret allows users to pass sensitive information, such as passwords, authentication tokens, SSH keys, and database credentials, to containers. A secret can then be referenced when declaring a container definition, and read from within containers as environment variables or from a local disk

**JOB:**

A job is used to create one or more pods and ensure that a specified number of them successfully terminate. It tracks the successful completions, and when a specified number of successful completions is reached, the job itself is complete. There are several types of jobs, including non-parallel jobs, parallel jobs with a fixed completion count, and parallel jobs with a work queue. A job should be used instead of a replication controller if you need to spread pods across cluster nodes and ensure, for example, so that each node has only one running pod of the specified type

**DAEMON SET:**

A daemon set ensures that all or some nodes run a copy of a pod. A daemon set tracks the additional and removal of cluster nodes and adds pods for nodes that are added to the cluster, terminates pods on nodes that are being removed from a cluster. Deleting a daemon set will clean up the pods it created. A typical use case for a daemon set is running a log collection daemon or a monitoring daemon on each node of a cluster

**QUOTA:**

A quota sets resource limitations, such as CPU, memory, number of pods or services, for a given namespace. It also forces users to explicitly request resource allotment for their pods

**NAMESPACE:**

A namespace provides a logical partition of the cluster’s resources. Kubernetes resources can use the same name when found in different namespaces. Different namespaces can be assigned different quotas for resource limitations