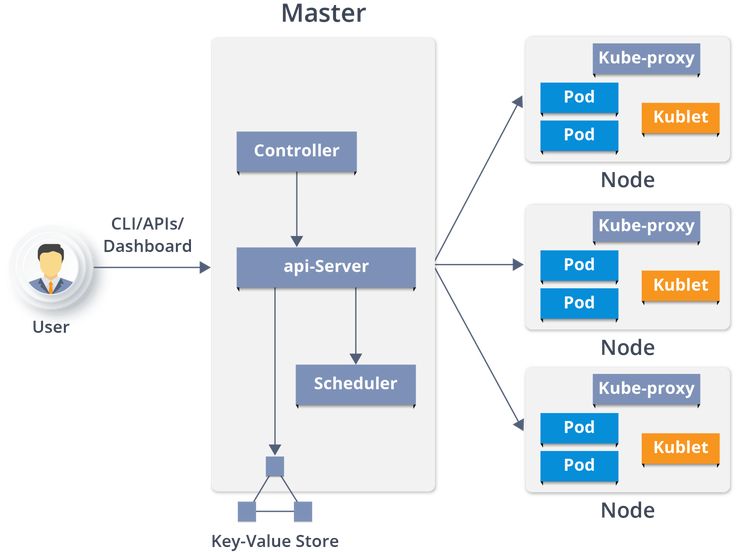
**Kubernetes k8s**

* K8S is a container orchestration technology that creates, deploy and manages clusters(bunch of docker containers)
* It schedules, runs and manages isolated containers which are running on virtual/physical/cloud machines , Convert isolated containers running on different H/W into cluster
* All 3 clouds support Kubernetes
* K8S originated at Google
* Features of Kubernetes
* Orchestration (Clustering of any no of Containers running on different H/W)
* Auto-Scaling (more clients? More demand)
* Auto- Healing (new containers in place of crashed containers)
* Load-Balancing (Distribute client requests)
* Platform Independent (Cloud/Virtualization/Physical

# **How to create an EKS cluster using AWS Console | Create node group | Configure Kubernetes cluster**

A cluster in Kubernetes (K8s) is a set of nodes (machines, either virtual or physical) that run containerized applications. A Kubernetes cluster manages the lifecycle of these containers and orchestrates their deployment, scaling, and operations.

A Kubernetes cluster consists of master nodes that control the cluster, and worker nodes that run the applications. It uses various components and abstractions like pods, services, and controllers to manage the lifecycle of containerized applications, ensuring they are deployed, scaled, and maintained properly.



Here are the main components of a Kubernetes cluster:

**1. Master Node**

The master node is responsible for managing the Kubernetes cluster. It contains the control plane components that handle all the decisions about the cluster (e.g., scheduling, scaling) and respond to cluster events.

* **API Server**: Exposes the Kubernetes API. The primary entry point for the Kubernetes control plane.
* **etcd**: A key-value store used to store all cluster data.
* **Scheduler**: Assigns work (pods) to the worker nodes.
* **Controller Manager**: Runs controller processes that handle routine tasks (e.g., replicating pods, handling node failures).

**2. Worker Nodes**

Worker nodes run the applications. They contain the components needed to run the containers and ensure they are healthy.

* **kubelet**: An agent that runs on each node. It ensures containers are running in pods.
* **kube-proxy**: Maintains network rules on nodes. It enables communication to and from pods.
* **Container Runtime**: Software responsible for running the containers (e.g., Docker, containerd).

**3. Pods**

Pods are the smallest deployable units in Kubernetes. A pod encapsulates one or more containers, storage resources, a unique network IP, and options that govern how the container(s) should run.

**4. Services**

Services are an abstraction that defines a logical set of pods and a policy by which to access them. Services enable load balancing and expose your application to the network.

**5. Controllers**

Controllers ensure that the desired state of the cluster matches the actual state. They are responsible for various aspects such as replication, endpoint management, and resource management.

* **ReplicationController/ReplicaSet**: Ensures that a specified number of pod replicas are running.
* **Deployment**: Manages stateless applications.
* **StatefulSet**: Manages stateful applications.
* **DaemonSet**: Ensures that a copy of a pod runs on all or some nodes.
* **Job**: Manages batch jobs.
* **CronJob**: Manages scheduled jobs.

**6. Namespaces**

Namespaces are a way to divide cluster resources between multiple users. They provide a scope for names, allowing you to create resources with the same name in different namespaces.

**7. ConfigMaps and Secrets**

* **ConfigMaps**: Used to store configuration data.
* **Secrets**: Used to store sensitive data such as passwords and tokens.

**8. Ingress**

Ingress is an API object that manages external access to the services in a cluster, typically HTTP.

creating an AWS EKS (Elastic Kubernetes Service) cluster using Terraform. Whether you're a beginner or an experienced user, our step-by-step guide provides detailed instructions and insights to help you set up your own EKS cluster with ease.

kubectl installation link:

# [Installing or updating kubectl - Amazon EKS](https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html) **How to create an EKS cluster using AWS Console | Create node group | Configure Kubernetes cluster**

**To install or update kubectl**

1. Determine whether you already have kubectl installed on your device.

**kubectl version --client**

1. Install or update kubectl on macOS, Linux, and Windows operating systems.

###### To install or update kubectl on Linux (amd64)

1. Download the kubectl binary for your cluster's Kubernetes version from Amazon S3.
   * Kubernetes 1.30

curl -O <https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/linux/amd64/kubectl>

curl -O <https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/linux/amd64/kubectl>

1. After you install kubectl, you can verify its version.

**kubectl version –client**

When first installing kubectl, it isn't yet configured to communicate with any server. We will cover this configuration as needed in other procedures. If you ever need to update the configuration to communicate with a particular cluster, you can run the following command. Replace *region-code* with the AWS Region that your cluster is in. Replace *my-cluster* with the name of your cluster.

aws eks update-kubeconfig --region region-code --name my-cluster

Copy the binary to a folder in your PATH. If you have already installed a version of kubectl, then we recommend creating a $HOME/bin/kubectl and ensuring that $HOME/bin comes first in your $PATH.

mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH

command for EKS cluster setup in kubectl server:

aws eks --region ap-southeast-1 describe-cluster --name pc-eks --query cluster.status

aws eks --region ap-southeast-1 update-kubeconfig --name pc-eks