Kubernetes Architecture

**Pod** - A Kubernetes pod is a collection of one or more containers and is the smallest unit of a Kubernetes application

**Deployment** – Offers auto-scaling and auto-healing functionality, which cannot be done through pods.

**Replica Set** – This carries out the auto-scaling and auto-healing functionality through the controller. This ensures the deployment file's desired user specification is always carried out.

**Service** – Services in Kubernetes take care of Kubernetes deployment with auto-scaling and auto-healing functionality constant IP changes after the failure of containers, and it does this through three steps which are:

* Load Balancing – Services provide the users with a load balancer instead of an IP address, which is not always constant
* Service Discovery – The service offers a selector and label in other to take care of the failure of IP address frequent changes when a container experiences any failure, and instead of service routing the user to a different IP address. It routes the user to the label and selector provided by the user, which is always constant.

Different types of service modes in Kubernetes:

* Cluster IP Mode – Only users who can access and SSH into your Kubernetes cluster can access your application such as fellow DevOps Engineers within the company
* Node Port Mode – The application can only be accessible by users that have access to your IP address or users within your organization
* Load balancer Mode – The application can be accessible over the internet

**K8s architecture**

* Control plane (master node)
  + API server - the core component of k8s, accepts all incoming requests, and exposes k8s to the external world.
  + etcd - key-value store, cluster-related information.
  + Scheduler - scheduling pods or resources on k8s, receives info from API server & acts on it
  + Controller manager - ensures controllers like replica set are running
  + Cloud controller manager - like Terraform
* Data plane (worker node)
  + Kubelet - creates pod, ensures pod is always running
  + Kube proxy - provides networking like Docker0, default load balancing
  + Container runtime - runs container inside pod

**What is the difference between Docker and Kubernetes**?

Docker - is an open-source containerization platform. In other words, Docker enables us to package applications into a container.

Kubernetes is a container orchestration platform that offers the advanced concept of auto-scaling, auto-healing, and enterprise-level support, which Docker as a platform lacks.

Ingress in Kubernetes

Why Ingress?

Before ingress in Kubernetes, people from other enterprise levels or have access to different Load Balancer customization using Nginx, F5, and enterprise. Which supports different customization of load balancing such as stickiness, ratio-based, path, domain, whitelisting, and blacklisting. Additionally, these were the features missing before Ingress:

* **Enterprise and TLS Load Balancer**
  + Sticky ness
  + TLS (HTTP, HTTPS)
  + Host (Domain)
  + Ratio-based load balancing
  + Whitelisting
  + Blacklisting
* **Load Balancer**
  + Cloud providers charge exorbitant amounts for every static IP address for the load balancer (Elastic IP addresses)

**Role Based Access Control (RBAC)**

This is for managing user permission and policies, and below is the following component:

* Service/user account
  + Use for managing users' permissions, and the service they can access (Pods, Service, Deployment, etc.)
* Roles/Cluster Roles
  + For assigning the respective user’s permissions
* Roles bindings/Cluster Roles
  + Assigned or attached the respective roles to the users, created by roles.