GKE components

- **Node -**

  Kubernetes runs your workload by placing containers into Pods to run on *Nodes*. A node may be a virtual or physical machine, depending on the cluster. Each node is managed by the control plane and contains the services necessary to run Pods.

- **Pod -**

  Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.
  A *Pod* (as in a pod of whales or pea pod) is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers.

- **Service -**

  A **Kubernetes service** is a logical abstraction for a deployed group of pods in a cluster (which all perform the same function).

  Since pods are ephemeral, a service enables a group of pods, which provide specific functions (web services, image processing, etc.) to be assigned a name and unique IP address (clusterIP). As long as the service is running that IP address, it will not change. Services also define policies for their access.

- **Ingress -**

  In Kubernetes, an Ingress is an object that allows access to your Kubernetes services from outside the Kubernetes cluster. You configure access by creating a collection of rules that define which inbound connections reach which services.

This lets you consolidate your routing rules into a single resource. For example, you might want to send requests to example.com/api/v1/ to an api-v1 service, and requests to example.com/api/v2/ to the api-v2 service. With an Ingress, you can easily set this up without creating a bunch of LoadBalancers or exposing each service on the Node.

- **ConfigMap -**

  A ConfigMap is an API object used to store non-confidential data in key-value pairs. Pods can consume ConfigMaps as environment variables, command-line arguments, or as configuration files in a volume.
  A ConfigMap allows you to decouple environment-specific configuration from your container images, so that your applications are easily portable.

- **Secrets -**

  A Secret is an object that contains a small amount of sensitive data such as a password, a token, or a key. Such information might otherwise be put in a Pod specification or in a container image. Using a Secret means that you don't need to include confidential data in your application code.

- **Volumes -**

  On-disk files in a container are ephemeral, which presents some problems for non-trivial applications when running in containers. One problem is the loss of files when a container crashes. The kubelet restarts the container but with a clean state. A second problem occurs when sharing files between containers running together in a Pod. The Kubernetes volume abstraction solves both of these problems.

- **Deployment -**

  A *Deployment* provides declarative updates for Pods and ReplicaSets.
  You describe a *desired state* in a Deployment, and the Deployment Controller changes the actual state to the desired state at a controlled rate. You can

define Deployments to create new ReplicaSets, or to remove existing Deployments and adopt all their resources with new Deployments.

- **StatefulSet -**

  StatefulSet is the workload API object used to manage stateful applications.
  Manages the deployment and scaling of a set of Pods, *and provides guarantees about the ordering and uniqueness* of these Pods.
  Like a Deployment, a StatefulSet manages Pods that are based on an identical container spec. Unlike a Deployment, a StatefulSet maintains a sticky identity for each of their Pods. These pods are created from the same spec, but are not interchangeable: each has a persistent identifier that it maintains across any rescheduling.