Master Node(s)

The master node consists of

- **etcd (distributed key-store):** It manages the Kubernetes cluster state. This distributed key-store can be a part of the master node or external to it. Nevertheless, all master nodes connect to it.
- **api server:** It manages all administrative tasks. The `api server` receives commands from the user (`kubectl` cli, REST or GUI); these commands are executed and the new cluster state is stored in the distributed key-store.
- **scheduler:** It schedules work to worker nodes by allocating pods. It is responsible for resource allocation.
- **controller:** It ensures that the desired state of the Kubernetes cluster is maintained. The desired state is what is contained in a JSON or YAML deployment file.

Worker Node(s)

The worker node(s) consists of

- **kubelet:** The `kubelet` agent runs on each worker node. It connects the worker node to the `api server` on the master node and receives instructions from it. It ensures the pods on the node are healthy.
- **kube-proxy:** It is the Kubernetes network proxy that runs on each worker node. It listens to the `api server` and forwards requests to the appropriate pod. It is important for load balancing.
- **pod(s):** It consists of one or more containers that share network and storage resources as well as container runtime instructions. Pods are the smallest deployable unit in Kubernetes.

Writing a Kubernetes Deployment File

The Kubernetes deployment file defines the desired state for the various Kubernetes objects. Examples of Kubernetes objects are

- **Pods:** It is a collection of one or more containers.

- **ReplicaSets:** It is part of the controller in the master node. It specifies the number of replicas of a pod that should be running at any given time. It ensures that the specified number of pods is maintained in the cluster.
- **Deployments:** It automatically creates `ReplicaSets`. It is also part of the `controller` in the master node. It ensures that the cluster's current state matches the desired state.
- **Namespaces:** It partitions the cluster into sub-clusters to organize users into groups.
- **Service:** It is a logical group of pods with a policy to access them.
    - *ServiceTypes:* It specifies the type of service, for example, ClusterIP, NodePort, LoadBalancer, and ExternalName. As an example, LoadBalancer exposes the service externally using a cloud provider's load balancer.

Other important tags in writing a Kubernetes deployment file

- **spec:** It describes the desired state of the cluster
- **metadata:** It contains information of the object
- **labels:** It is used to specify attributes of objects as key-value pairs
- **selector:** It is used to select a subset of objects based on their label values

The deployment file is specified as a yaml file.

Deploying Kubernetes on Google Kubernetes Engine

Google Kubernetes engine (GKE) provides a managed environment for deploying application containers. To create and deploy resources on GCP from the local shell, the Google command-line SDK gcloud will have to be installed and configured. If this is not the case on your machine, follow the instructions at `https://cloud.google.com/sdk/gcloud/`. Otherwise, a simpler option is to use the Google Cloud Shell which already has gcloud and kubectl (the Kubernetes command-line interface) installed.