Features of Kubernetes

The following are some features of Kubernetes:

- Horizontal auto-scaling: Dynamically scales containers based on resource demands
- Self-healing: Re-provisions failed nodes in response to health checks
- Load balancing: Efficiently distributes requests between containers in a pod
- Rollbacks and updates: Easily update or revert to a previous container deployment without causing application downtime
- DNS service discovery: Uses Domain Name System (DNS) to manage container groups as a Kubernetes service

Components of Kubernetes

The main components of the Kubernetes engine are

- Master node(s): Manages the Kubernetes cluster. There may be more than one master node in high availability mode for fault-tolerance purposes. In this case, only one is the master, and the others follow.
- Worker node(s): Machine(s) that runs containerized applications that are scheduled as pod(s).

The illustration in Figure 45-6 provides an overview of the Kubernetes architecture.

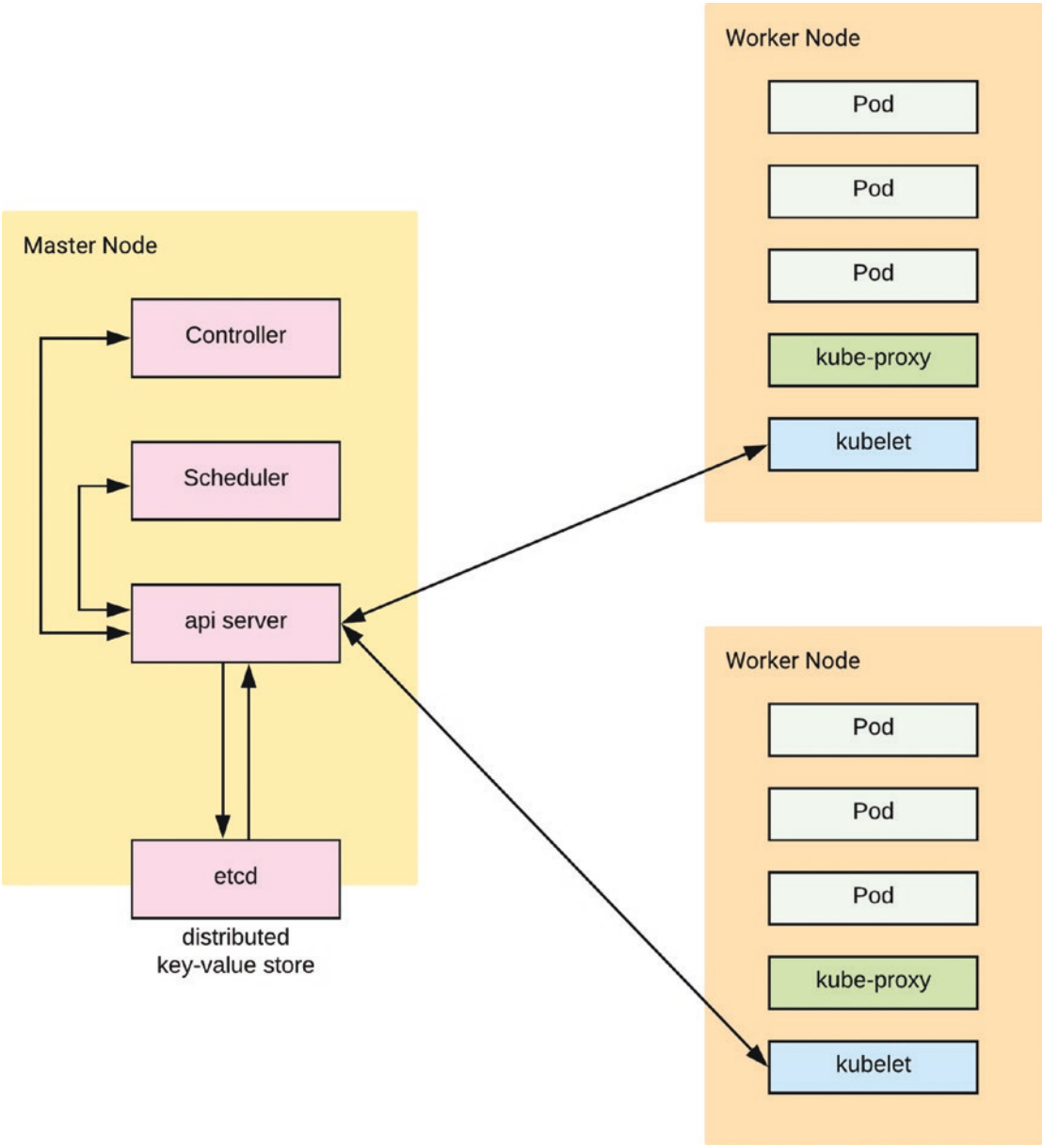

Figure 45-6. *High-level overview of Kubernetes components*

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.