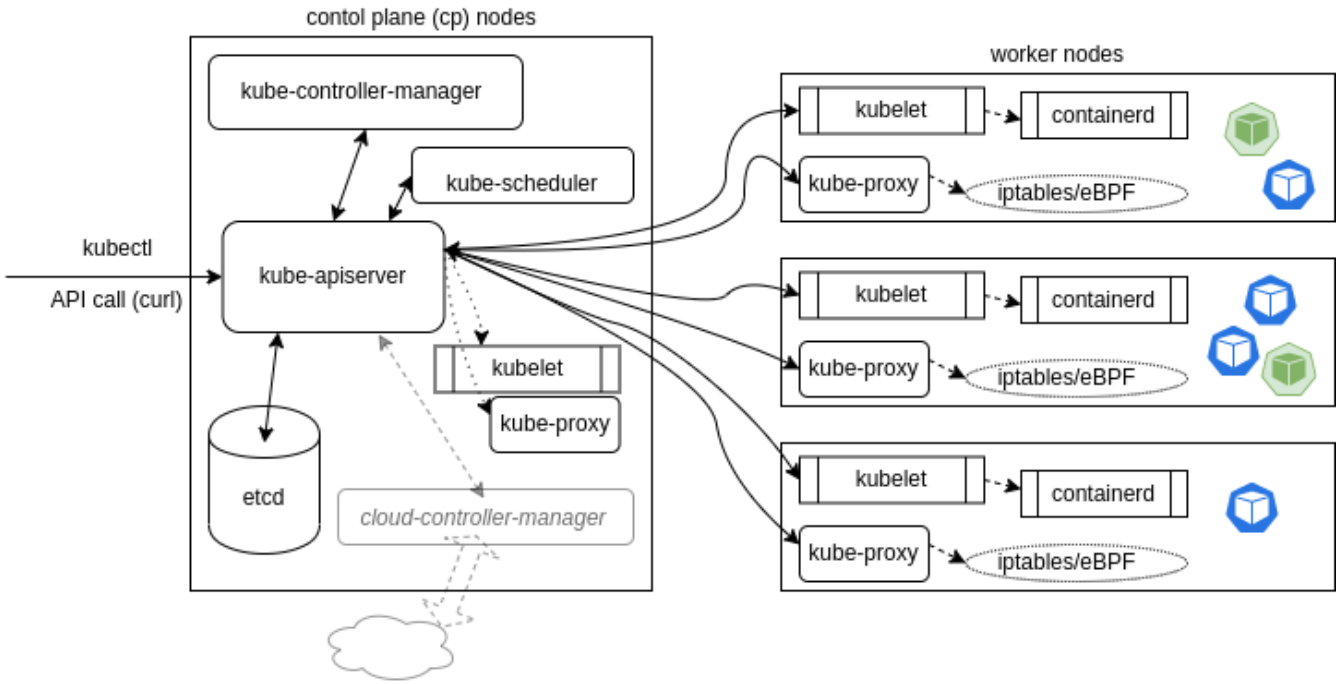
BASICS OF KUBERNETES

Basics of Kubernetes

Kubernetes Architecture

To quickly demystify Kubernetes, let's have a look at the *Kubernetes Architecture* graphic, which shows a high-level architecture diagram of the system components. Not all components are shown. Every node running a container would have **kubelet** and **kube-proxy**, for example.



Kubernetes Architecture

In its simplest form, Kubernetes is made of control plane nodes (aka **cp** nodes) and worker nodes, once called minions. We will see in a follow-on chapter how you can actually run everything on a single node for testing purposes. The cp runs an API server, a scheduler, various controllers and a storage system to keep the state of the cluster, container settings, and the networking configuration.

Kubernetes exposes an API via the API server. You can communicate with the API using a local client called **kubectl** or you can write your own client and use **curl** commands. The **kube-scheduler** is forwarded the pod spec for running containers coming to the API and finds a suitable node to run those containers. Each node in the cluster runs two processes: a kubelet, which is often a **systemd** process, not a container, and kube-proxy. The kubelet receives requests to run the containers, manages any resources necessary and works with the container engine to manage them on the local node. The local container engine could be Docker, cri-o, containerd, or some other.

The **kube-proxy** creates and manages networking rules to expose the container on the network to other containers or the outside world.

Using an API-based communication scheme allows for non-Linux worker nodes and containers. Support for Windows Server 2019 was graduated to *Stable* with the 1.14 release. Only Linux nodes can be cp of the cluster at this time.