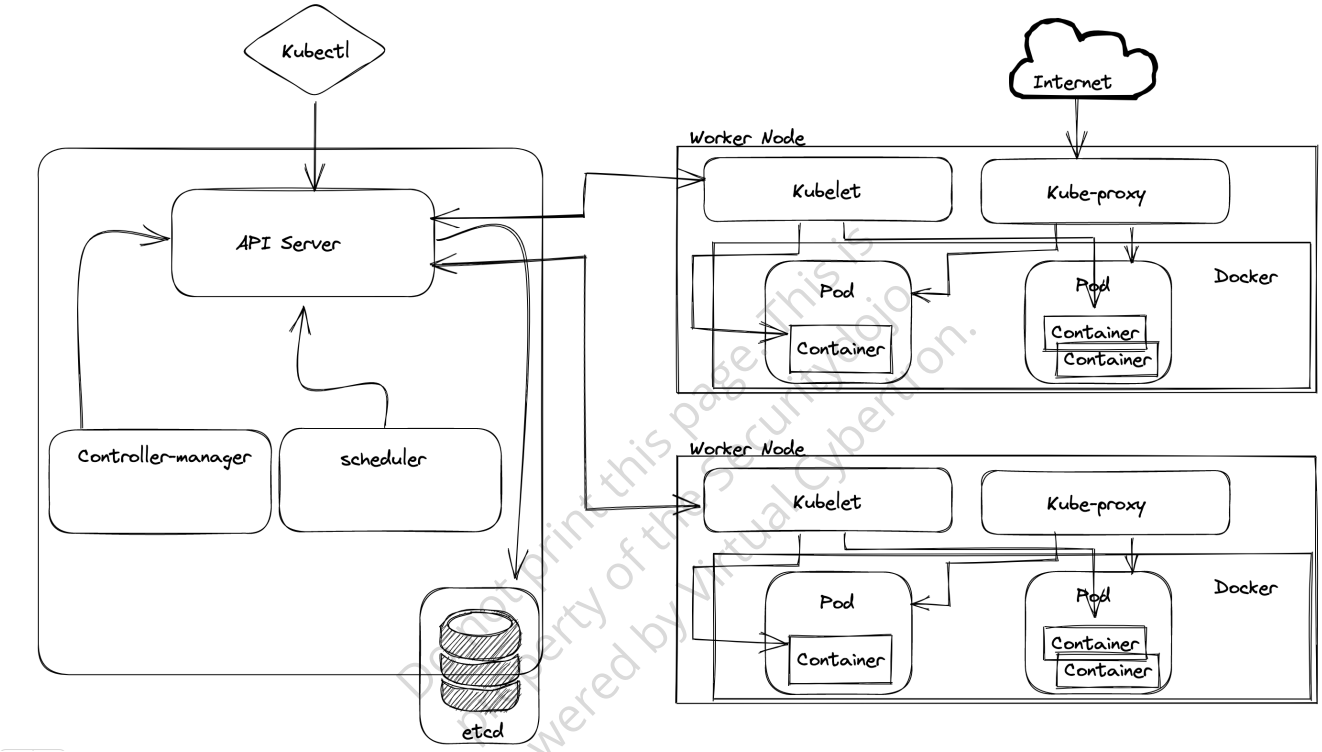
# Explanation of Key Kubernetes Components

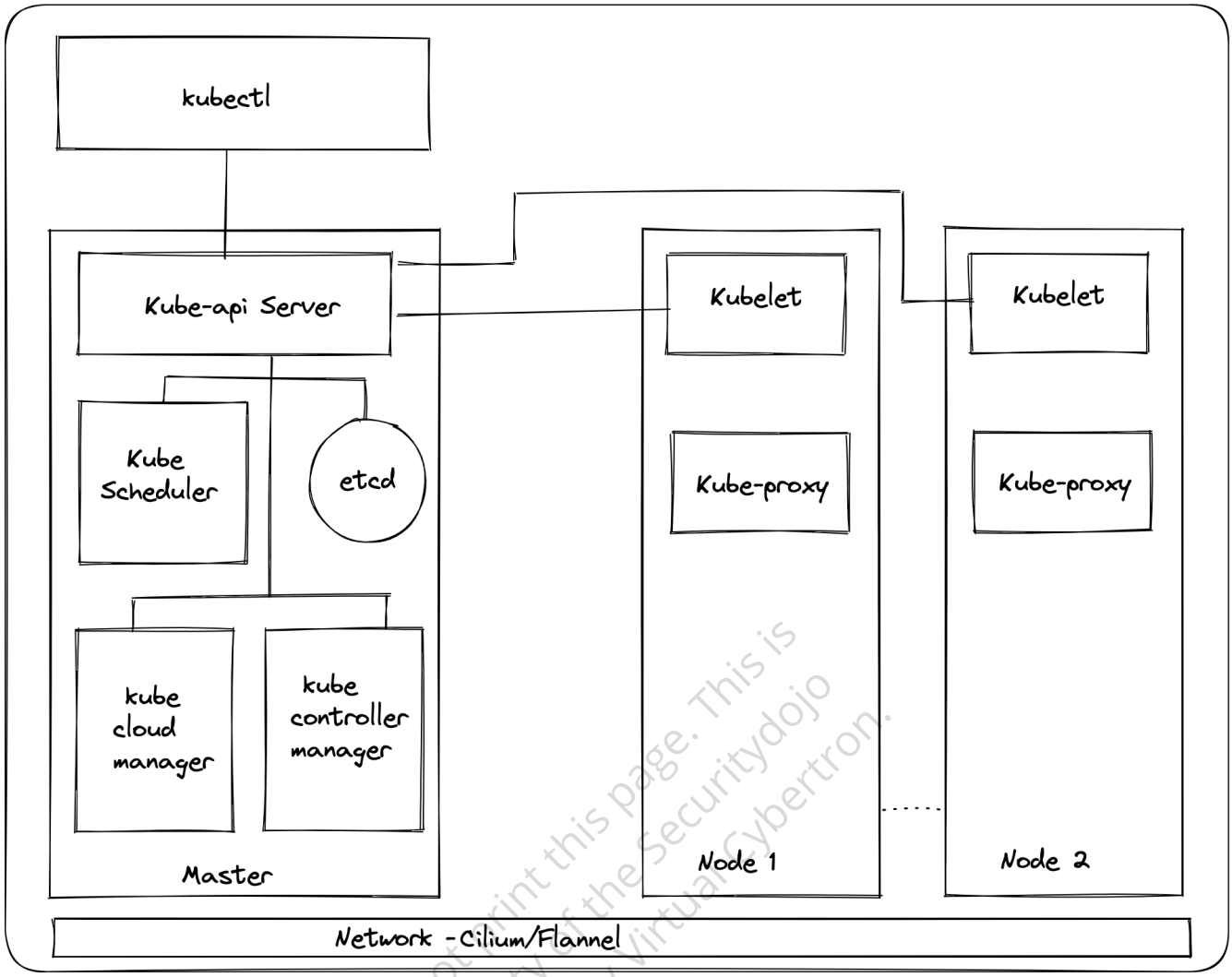
## Kubernetes Components and Architecture






- Kubernetes follows a client-server architecture. It's possible to have a multi-master setup (for high availability), but by default there is a single master server which acts as a controlling node and point of contact.
- The master server consists of various components including a kube-apiserver, an etcd storage, a kube-controller-manager, a cloud-controller-manager, a kube-scheduler, and a DNS server for Kubernetes services.
- Node components include kubelet and kube-proxy on top of Docker.

High level Kubernetes architecture diagram showing a cluster with a master and two worker nodes:






## Master Components

Below are the main components found on the master node:

1. **Kube API Server:**

- This component is responsible for accepting and processing commands that view or modify the state of the cluster, including the launch of pods.
- The API server verifies the authenticity and authorization of requests, manages admission control, and connects with the kubelet.

2. **Etcd**:

- A simple, distributed key value storage which is used to store the Kubernetes cluster data such as number of pods, their state, namespace, etc as well as service discovery details.
- It is only accessible from the API server for security reasons.

3. **Kube-scheduler**:

- The scheduler's role is to assign pods to nodes and evaluate the requirements of each pod. After the evaluation, the scheduler selects the most suitable node for the pod.
- The scheduler does not launch the pods, but instead writes the node name into the pod object.
- The scheduler also keeps track of all pod states and handles hardware, software, and policy constraints.

4. **Kube-controller-manager**: The controller manager continuously monitors the state of the cluster through the kube API server and makes changes to achieve the desired state if the current state does not match. The controller manager communicates important information, such as if a node goes offline. The manager is called as such because it maintains Kubernetes objects through loops of code called controllers.

5. **Cloud-controller-manager (optional)**:

- The cloud controller manager runs controllers that interact with the underlying cloud providers, allowing the code for the cloud vendor and Kubernetes to evolve independently.
- The cloud controller manager is responsible for features such as load balancing and storage volumes as needed.
- Prior to its introduction, the core Kubernetes code was dependent on cloud-provider-specific code for functionality.



## Node (worker) components:

Below are the main components found on a (worker) node:

1. **Kubelet**: the main service on a node, regularly taking in new or modified pod specifications (primarily through the kube-apiserver) and ensuring that pods and their containers are healthy and running in the desired state. This component also reports to the master on the health of the host where it is running.

2. **Kube-proxy**:

- A proxy service that runs on each worker node to deal with individual host subnetting and expose services to the external world.
- It performs request forwarding to the correct pods/containers across the various isolated networks in a cluster.

3. **Kubernetes Pod**:

- The smallest and simplest unit in the Kubernetes object model that encapsulates an application container.

3. **Kubectl**:

- kubectl command is a line tool that interacts with kube-apiserver and send commands to the master node. Each command is converted into an API call.

---

For further reading, see Kubernetes Documentation: Kubernetes Components

---