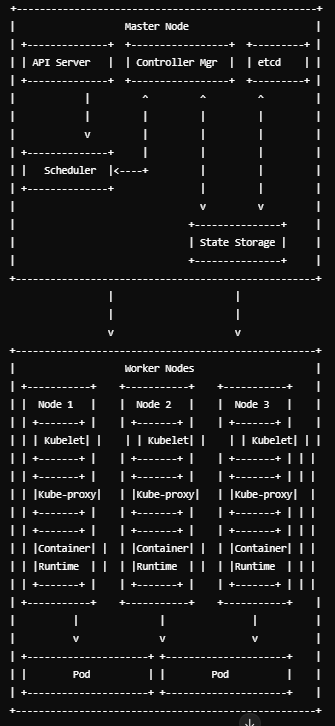
**Kubernetes Architecture Components**

1. **Master Node**: Manages the Kubernetes cluster.
   * **API Server**: Serves the Kubernetes API using JSON over HTTP, which is the front end of the Kubernetes control plane.
   * **Controller Manager**: Runs controller processes to handle routine tasks, ensuring the desired state of the cluster matches the current state.
   * **Scheduler**: Assigns newly created pods to nodes based on resource requirements and other constraints.
   * **etcd**: A key-value store used for configuration data, service discovery, and state management.
2. **Worker Nodes**: Execute the containerized applications.
   * **Kubelet**: An agent that runs on each worker node, ensuring containers are running as expected.
   * **Kube-proxy**: Manages network connectivity and load balancing for services in the cluster.
   * **Container Runtime**: Software responsible for running containers (e.g., Docker, containerd).
   * **Pods**: The smallest deployable units in Kubernetes, which can contain one or more containers.

**Diagram**

Here is a simplified diagram to illustrate the architecture:



**Key Points:**

* **Master Node**: Coordinates the entire cluster, managing workloads and maintaining desired states.
* **Worker Nodes**: Run the containerized applications and report to the master node.

This architecture allows Kubernetes to efficiently manage and scale containerized applications across various environments.

**Kubelet**

Kubelet is an agent that runs on each worker node in the Kubernetes cluster. Its primary responsibilities include:

1. **Pod Management**: Kubelet ensures that the containers described in PodSpecs are running and healthy. It periodically receives PodSpecs from the Kubernetes control plane (API server).
2. **Node Monitoring**: Kubelet monitors the state of the node and reports it to the Kubernetes API server. It tracks the status of running pods and the resources they consume.
3. **Execution Management**: Kubelet interacts with the container runtime (e.g., Docker, containerd) to start, stop, and maintain containers.
4. **Health Checking**: It performs liveness and readiness probes on running containers to ensure they are functioning correctly.

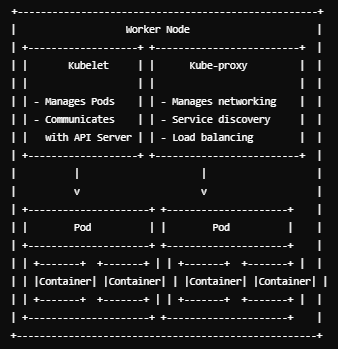
**Kube-proxy**

Kube-proxy is a network proxy that runs on each worker node. Its primary responsibilities include:

1. **Service Networking**: Kube-proxy maintains network rules on nodes to allow communication to your pods from network sessions inside or outside of your cluster.
2. **Service Discovery**: It enables the routing of network traffic to the appropriate pods based on the IP address and port number defined by Kubernetes services.
3. **Load Balancing**: Kube-proxy provides basic load balancing for network traffic across multiple pods. It uses various modes like user-space proxy mode, iptables mode, or IPVS mode to manage network traffic efficiently.

**Diagram Representation**

Here’s a simple representation of how kubelet and kube-proxy fit into the Kubernetes architecture:



**Summary**

* **Kubelet**: Manages the lifecycle of containers on a node, ensuring they are running as specified and reporting their status.
* **Kube-proxy**: Manages the networking aspects on a node, ensuring efficient and correct routing of traffic to and from pods, enabling service discovery and load balancing within the cluster.