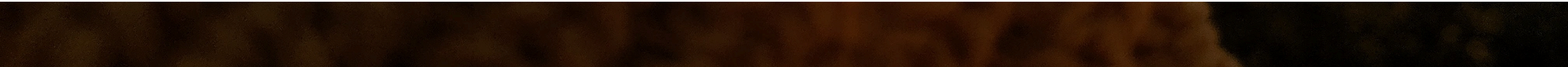
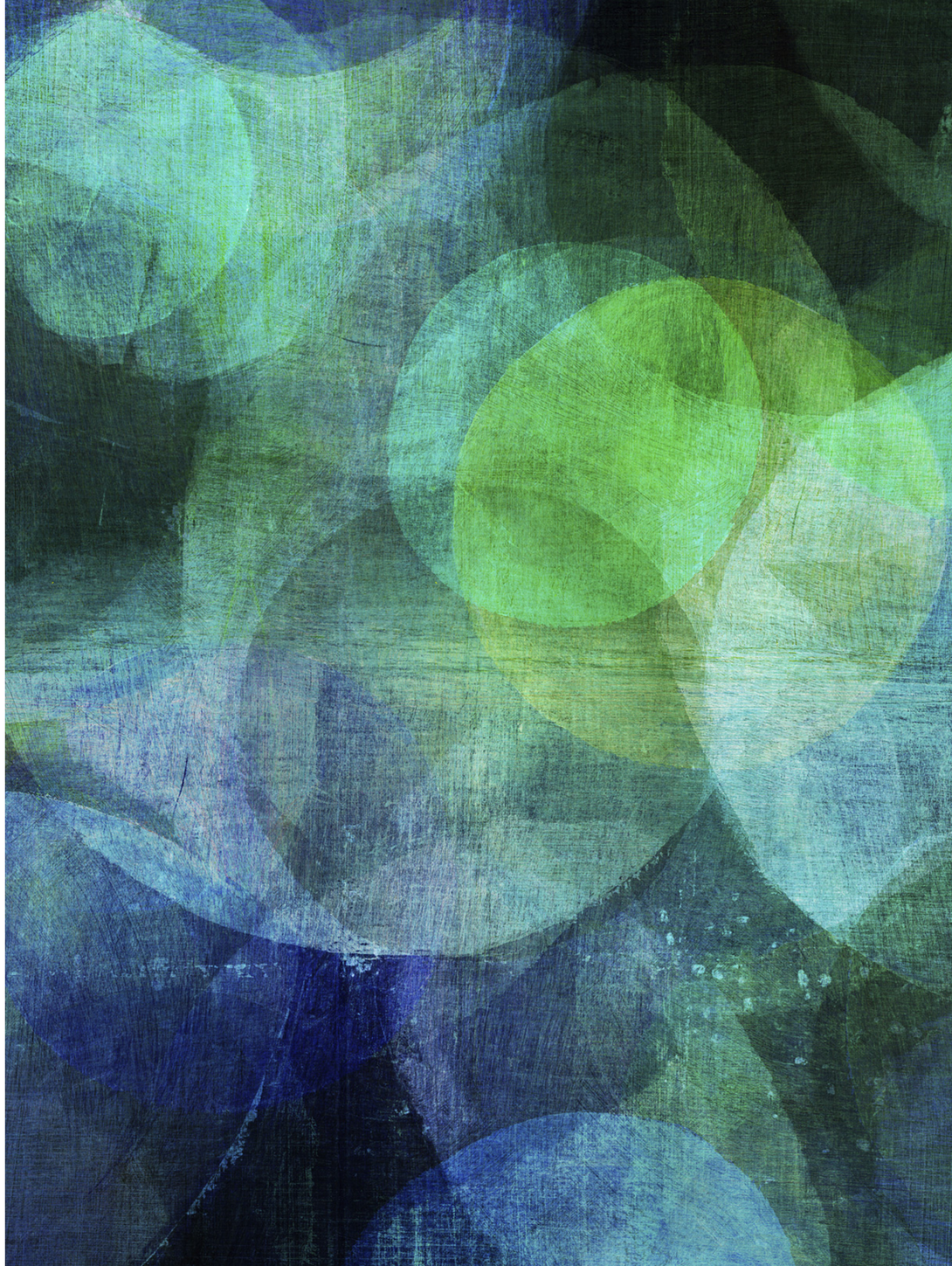
INTRODUCTION TO KUBERNETES

INTRODUCTION TO KUBERNETES

Kubernetes Architecture

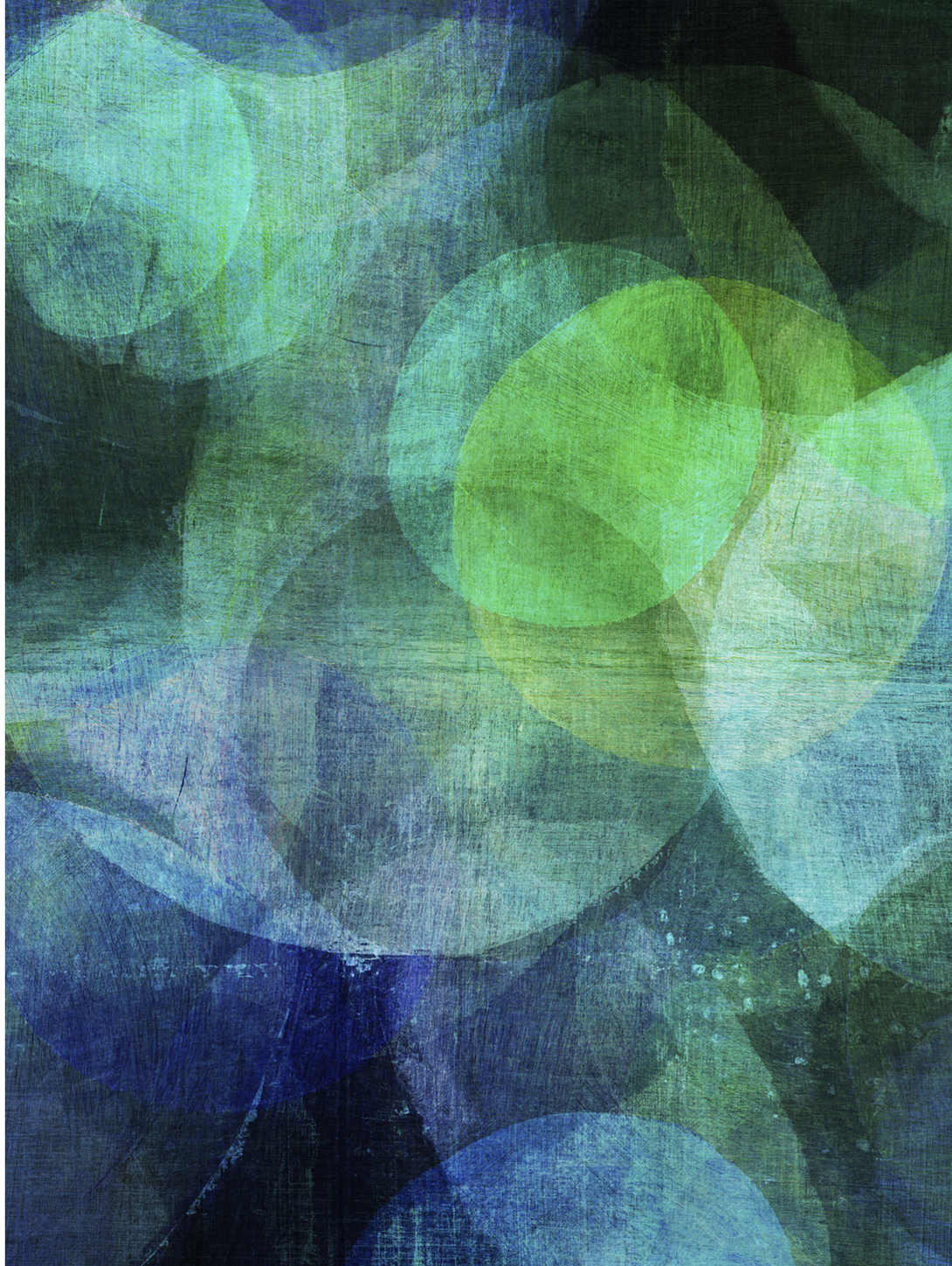


AGENDA



AGENDA

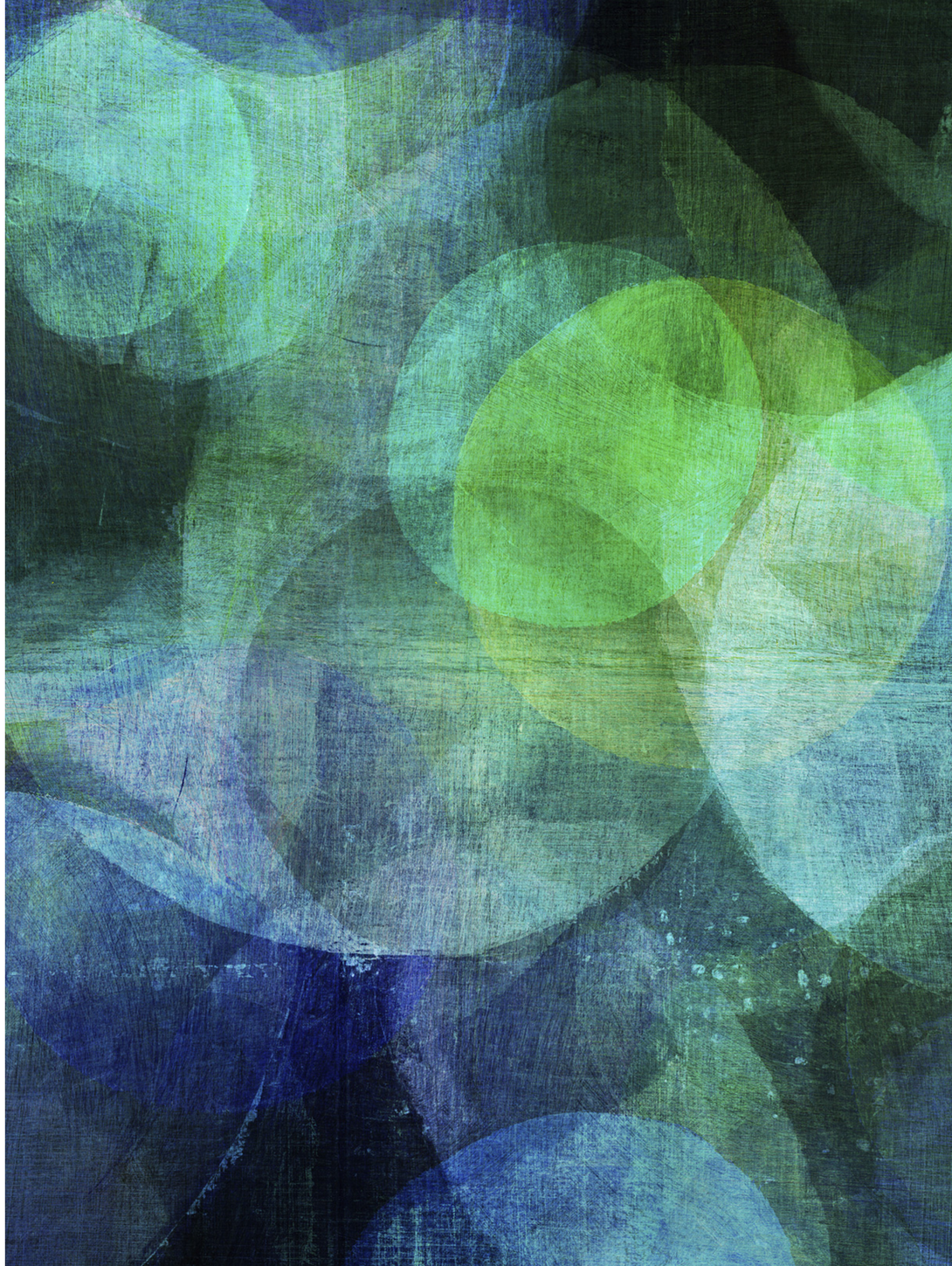
- Worker Nodes



AGENDA

- Worker Nodes

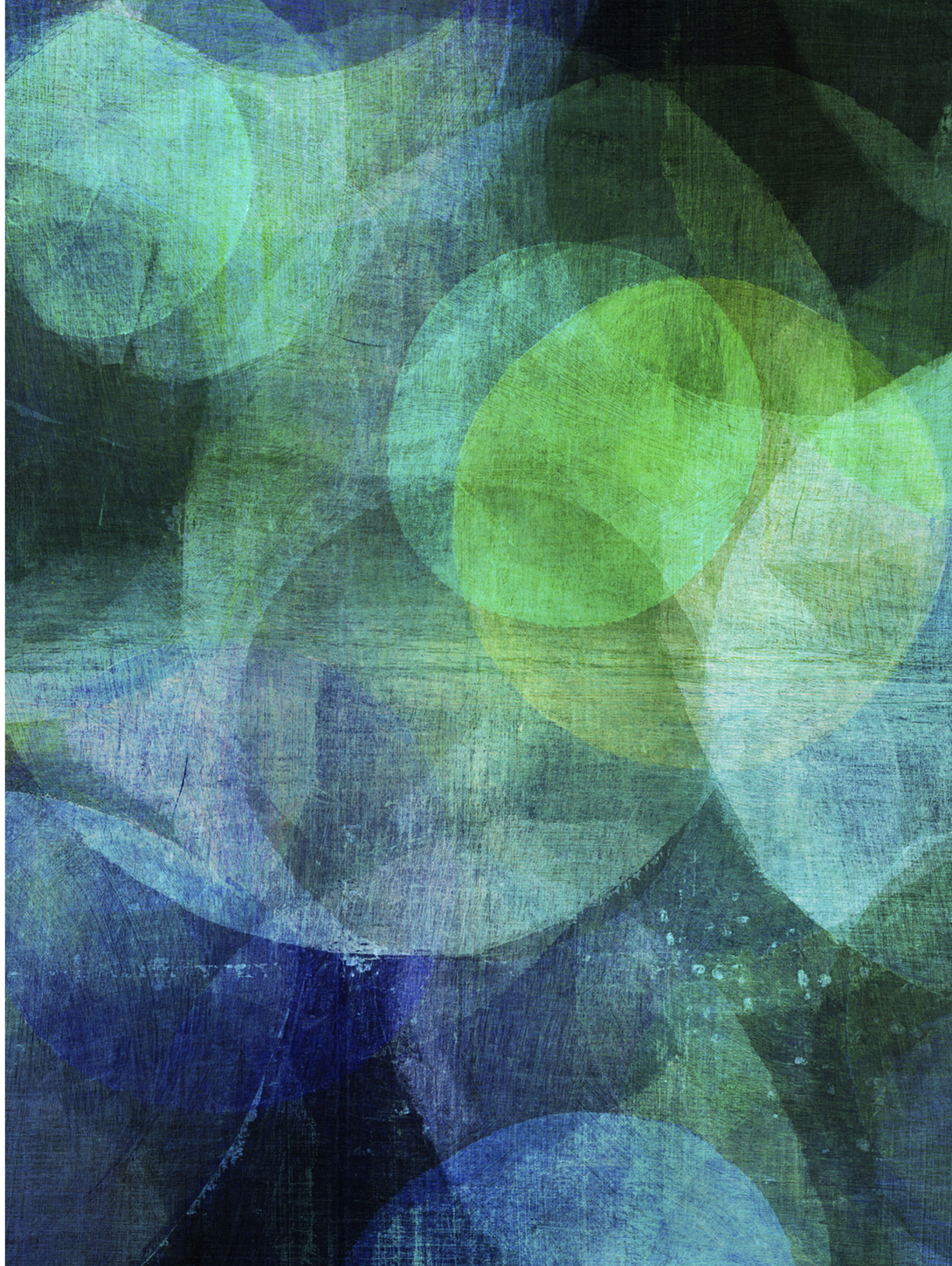
- Container Runtime



AGENDA

- Worker Nodes

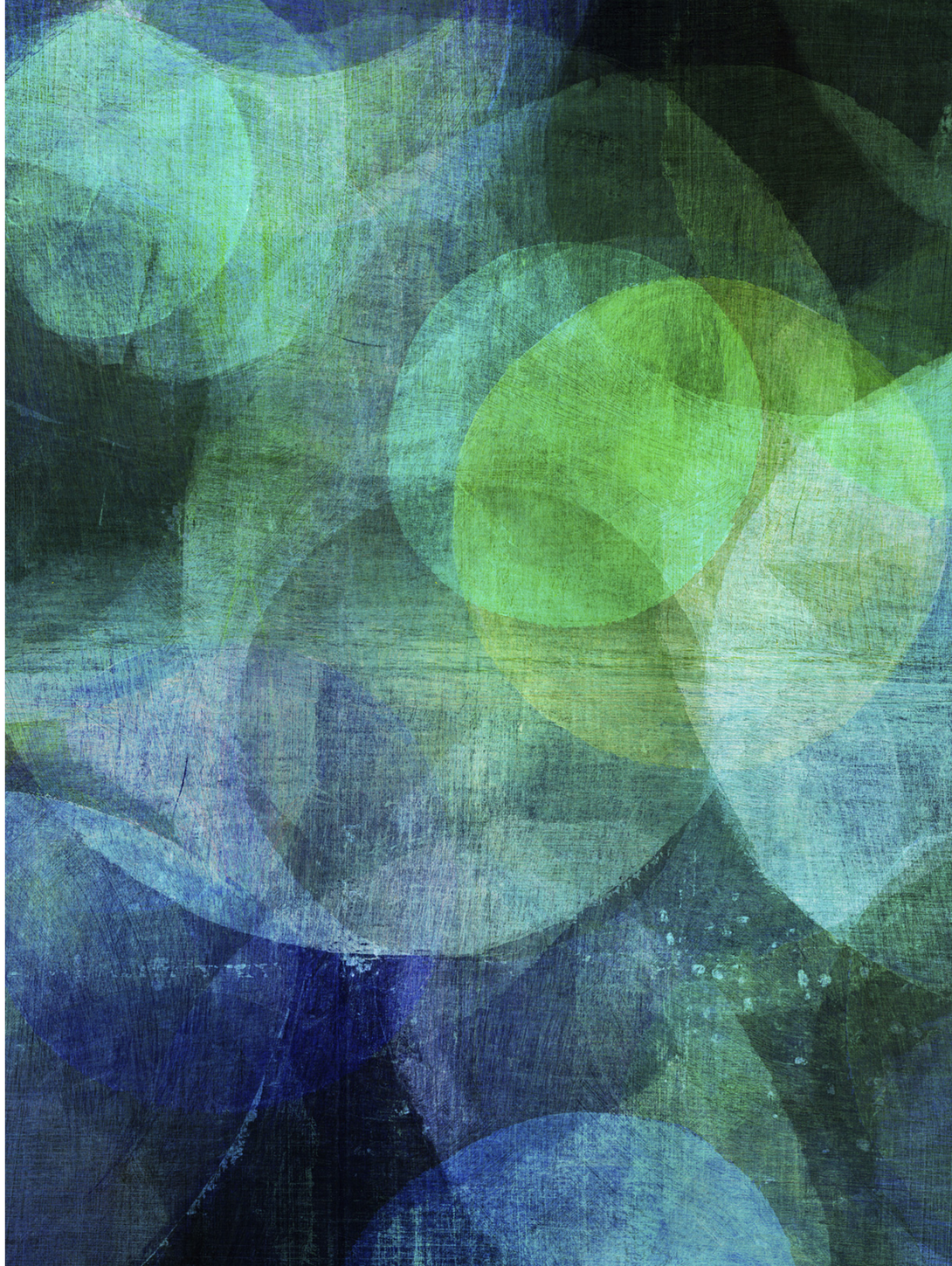
- Container Runtime
- Kubelet



AGENDA

- Worker Nodes

- Container Runtime
- Kubelet
- Kube Proxy

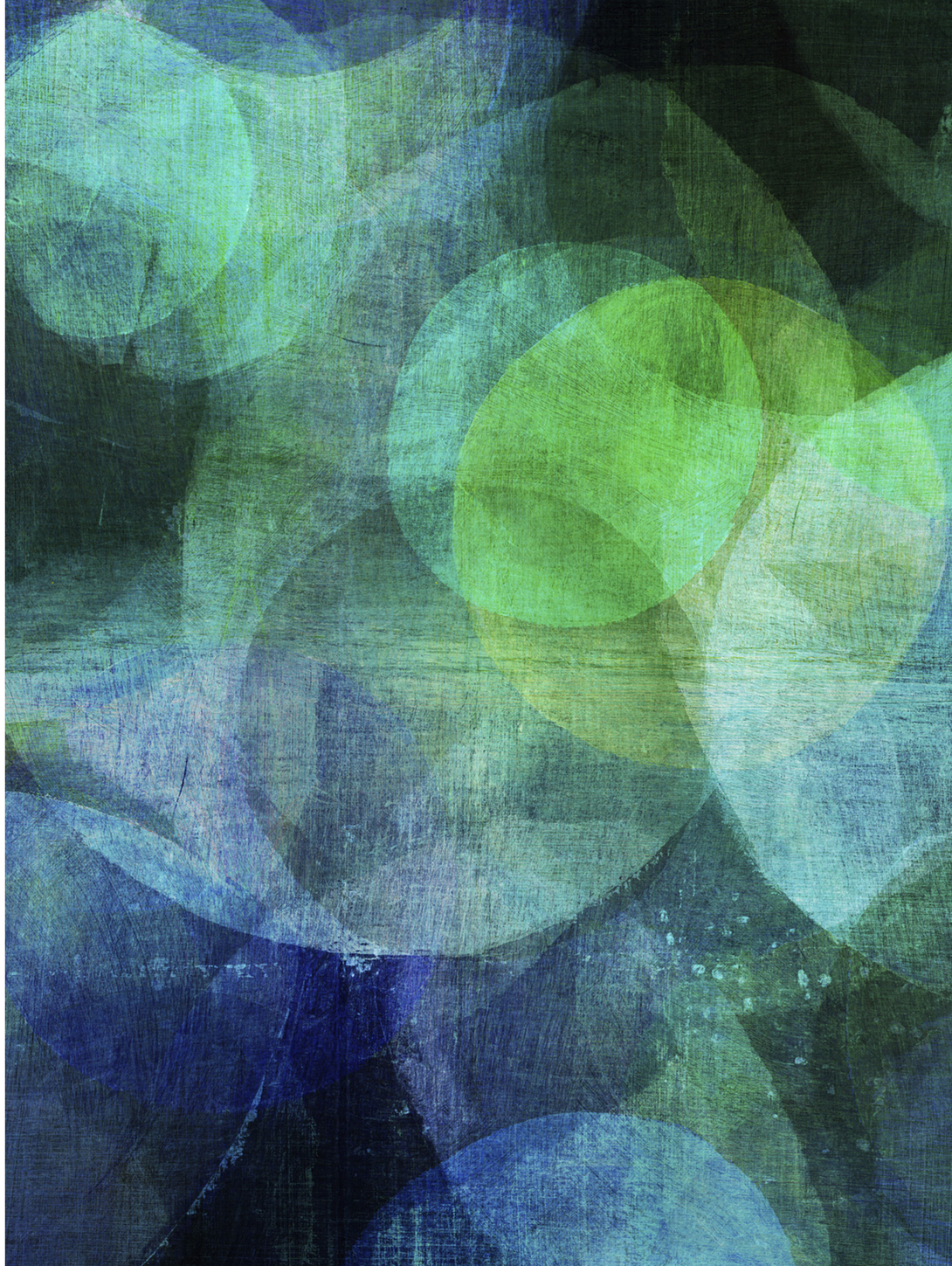


AGENDA

- Worker Nodes

- Container Runtime
- Kubelet
- Kube Proxy

- Master Nodes



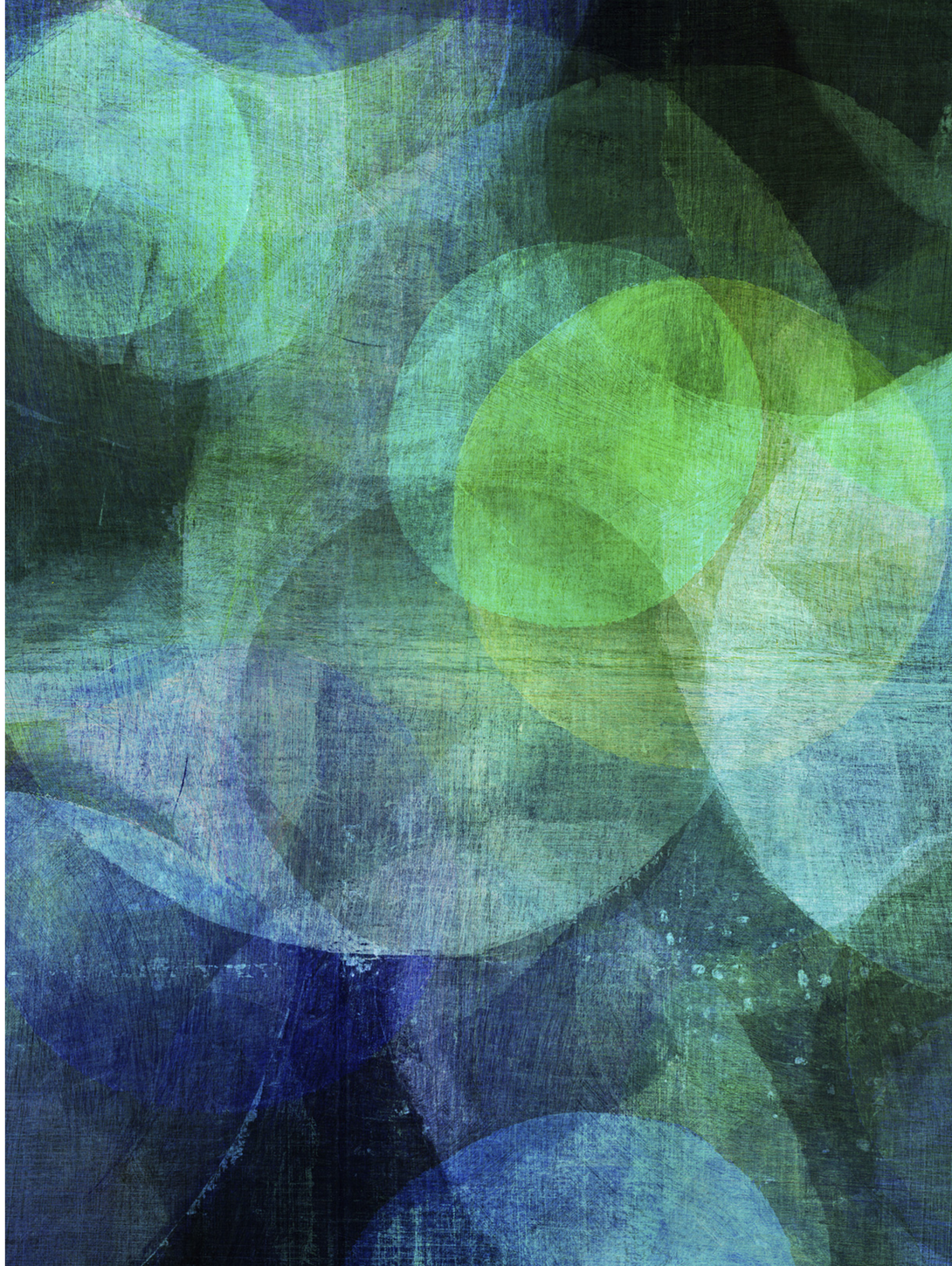
AGENDA

- Worker Nodes

- Container Runtime
- Kubelet
- Kube Proxy

- Master Nodes

- API Server



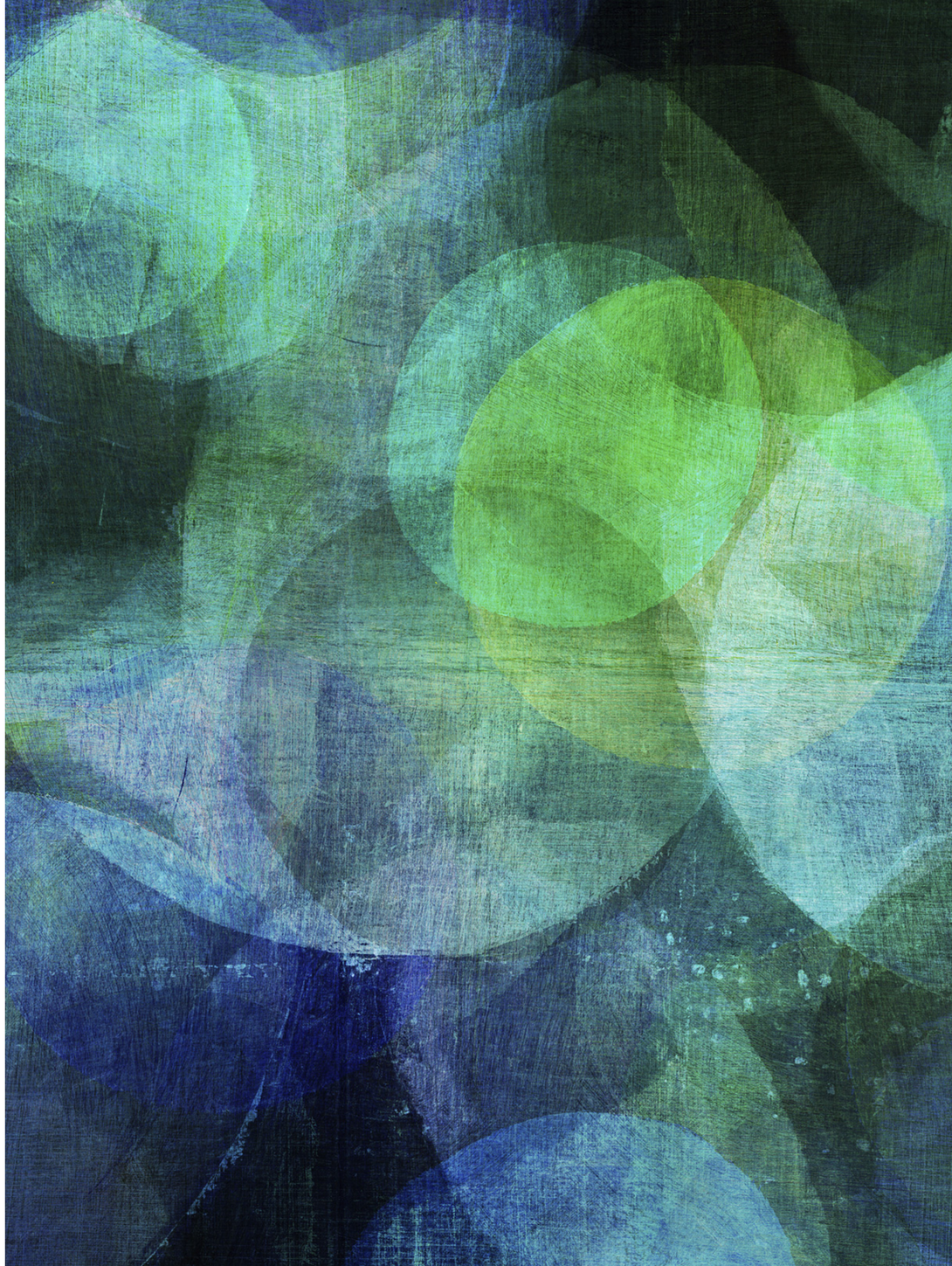
AGENDA

- Worker Nodes

- Container Runtime
- Kubelet
- Kube Proxy

- Master Nodes

- API Server
- Scheduler



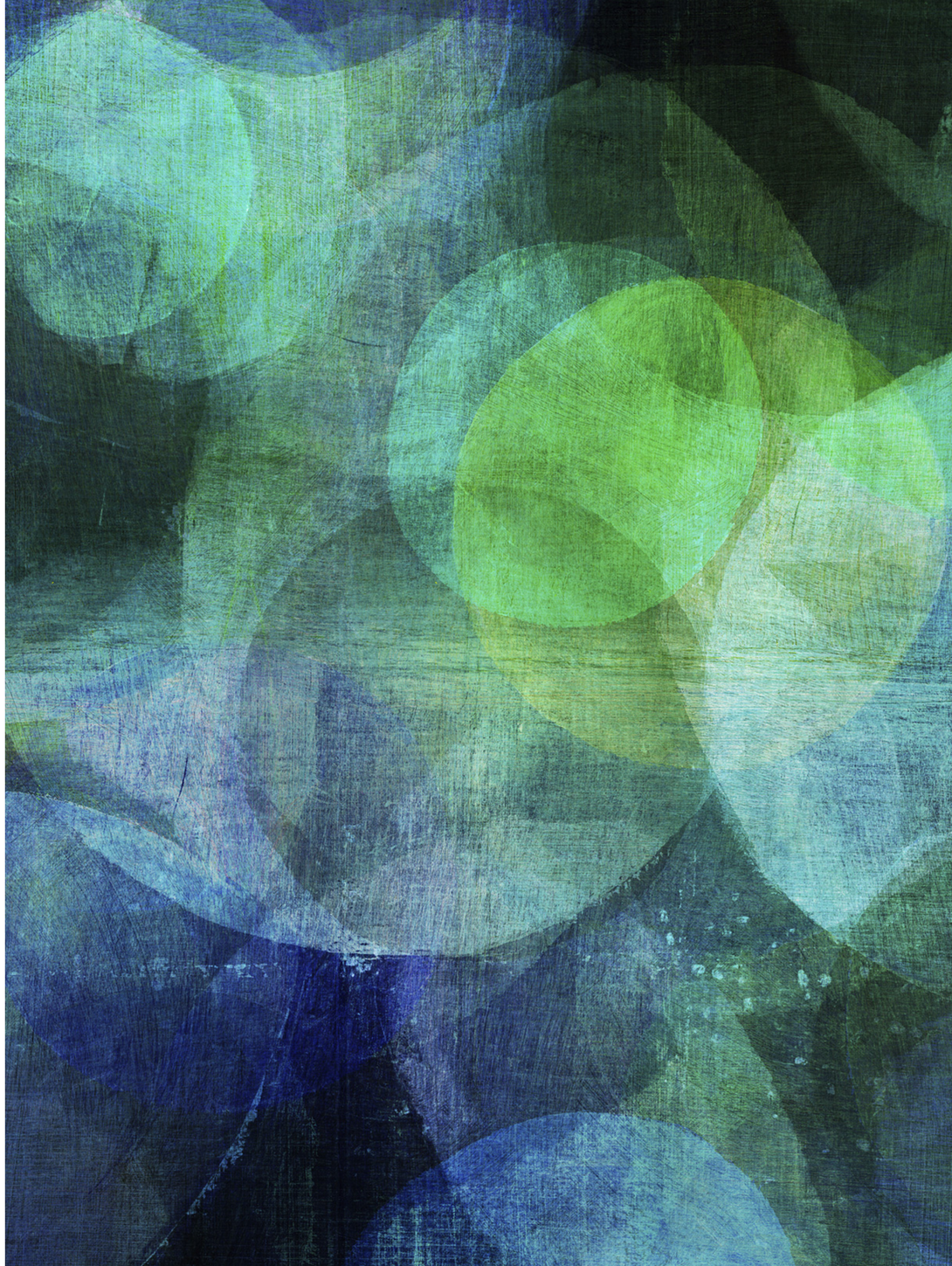
AGENDA

- Worker Nodes

- Container Runtime
- Kubelet
- Kube Proxy

- Master Nodes

- API Server
- Scheduler
- Control Manager



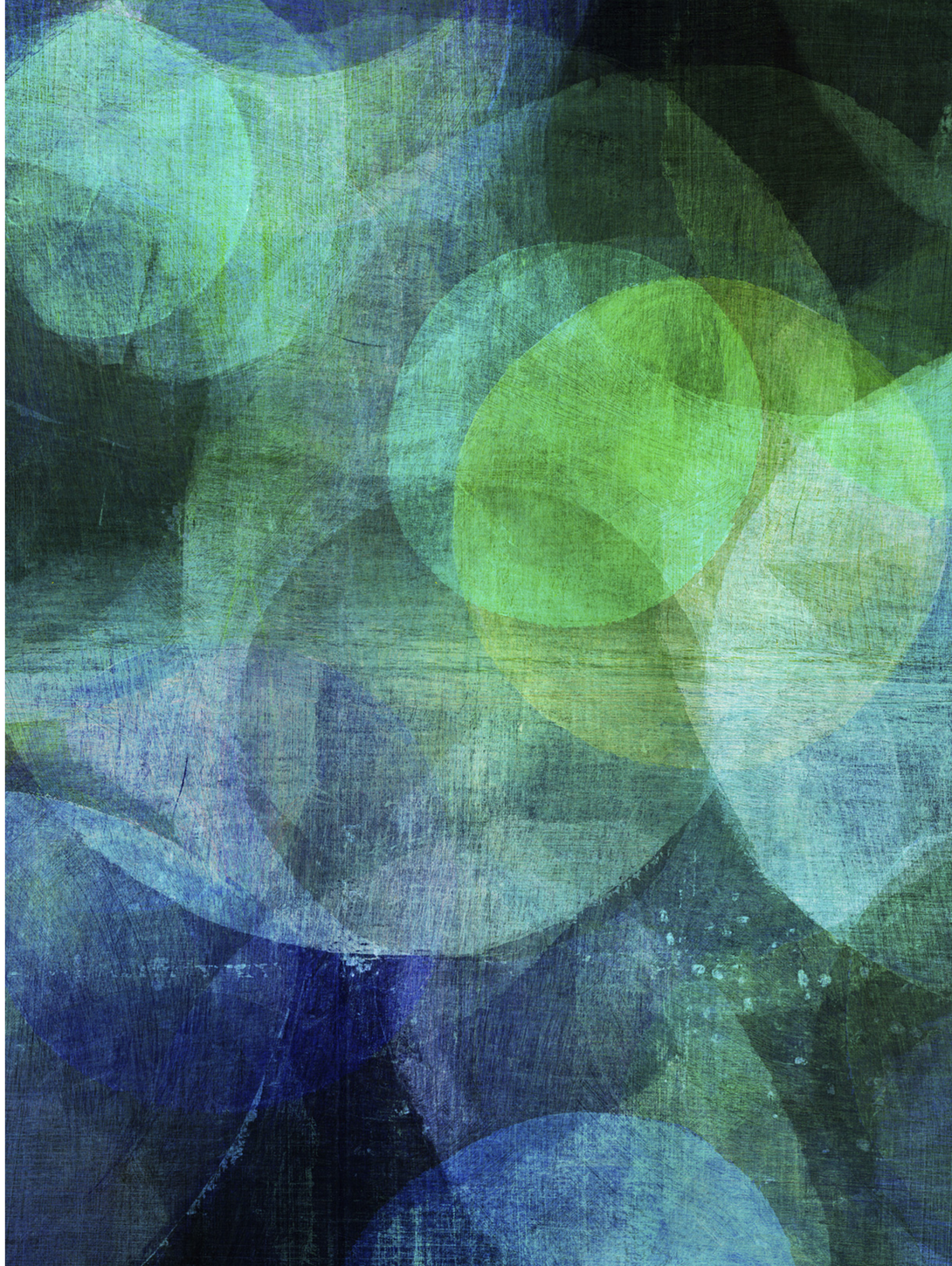
AGENDA

- Worker Nodes

- Container Runtime
- Kubelet
- Kube Proxy

- Master Nodes

- API Server
- Scheduler
- Control Manager
- ETCD



NODE COMPONENT

–

WORKER NODE

KUBERNETES ARCHITECTURE - NODE COMPONENT (WORKER NODE)

Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.

KUBERNETES ARCHITECTURE - NODE COMPONENT (WORKER NODE)

Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.

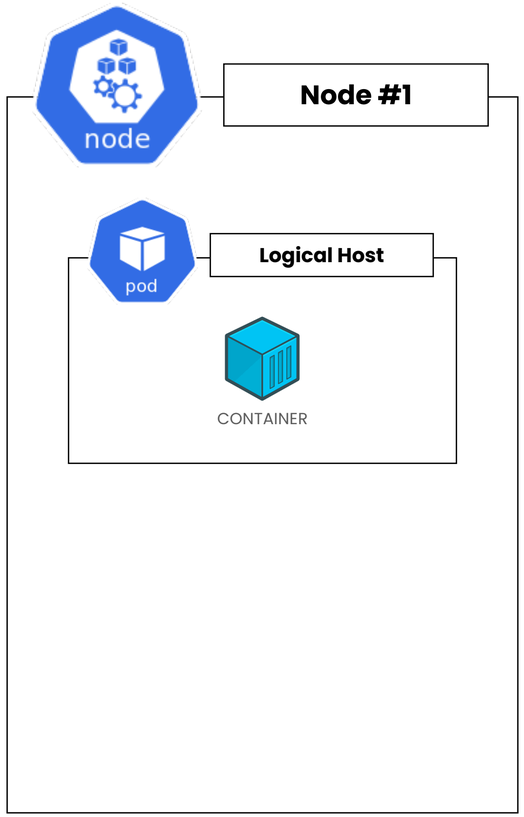
KUBERNETES ARCHITECTURE - NODE COMPONENT (WORKER NODE)

Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.

- Multi Pod in each node

KUBERNETES ARCHITECTURE - NODE COMPONENT (WORKER NODE)

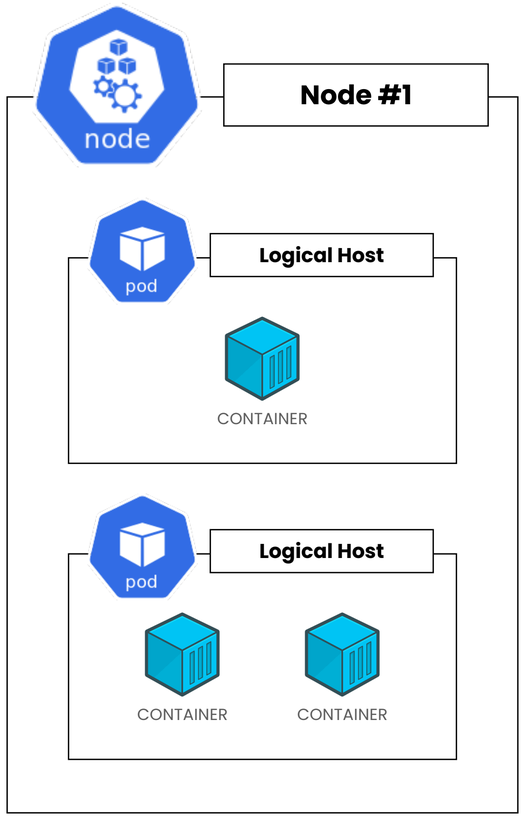
Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.



- Multi Pod in each node

KUBERNETES ARCHITECTURE - NODE COMPONENT (WORKER NODE)

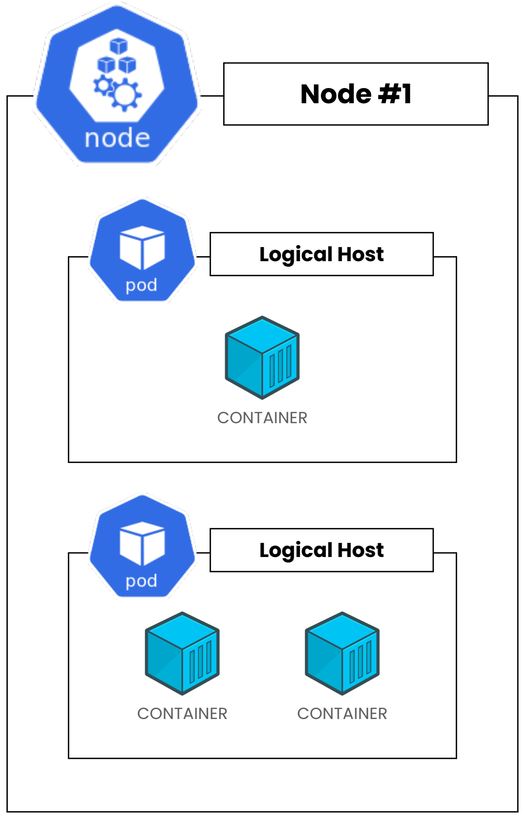
Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.



- Multi Pod in each node

KUBERNETES ARCHITECTURE - NODE COMPONENT (WORKER NODE)

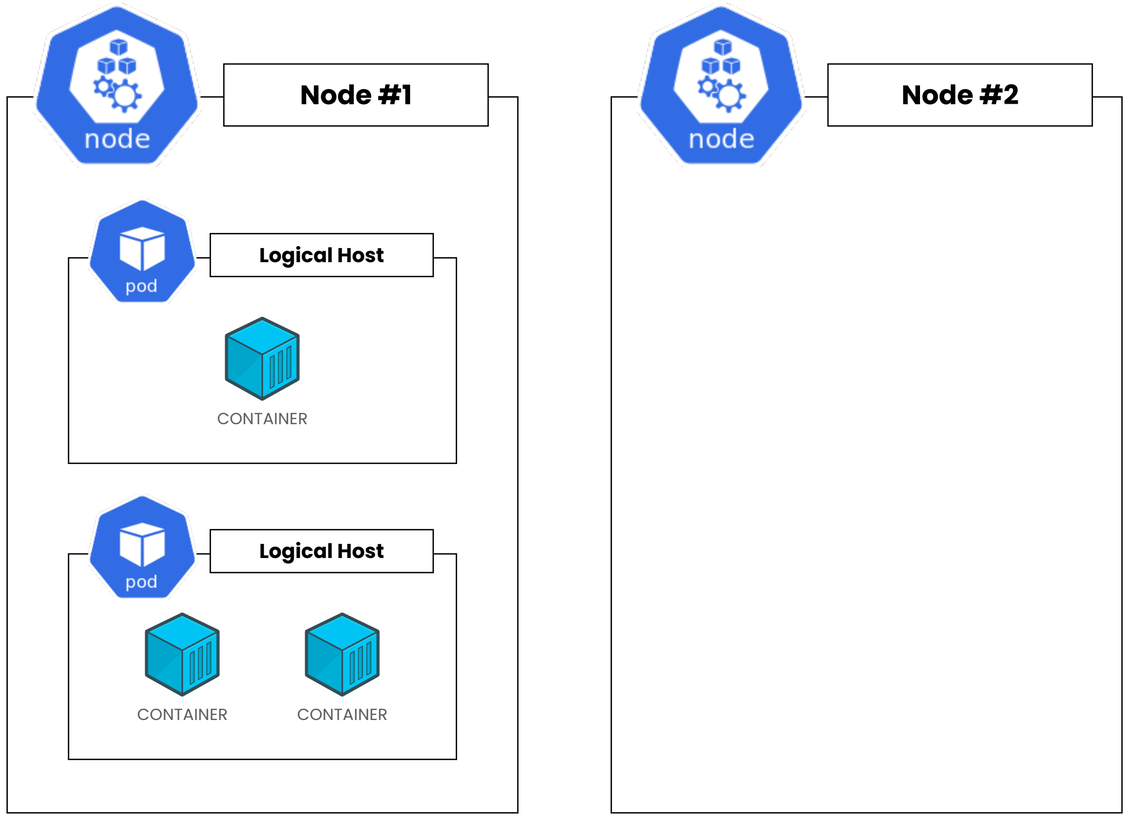
Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.



- Multi Pod in each node
- Doing the actual work

KUBERNETES ARCHITECTURE - NODE COMPONENT (WORKER NODE)

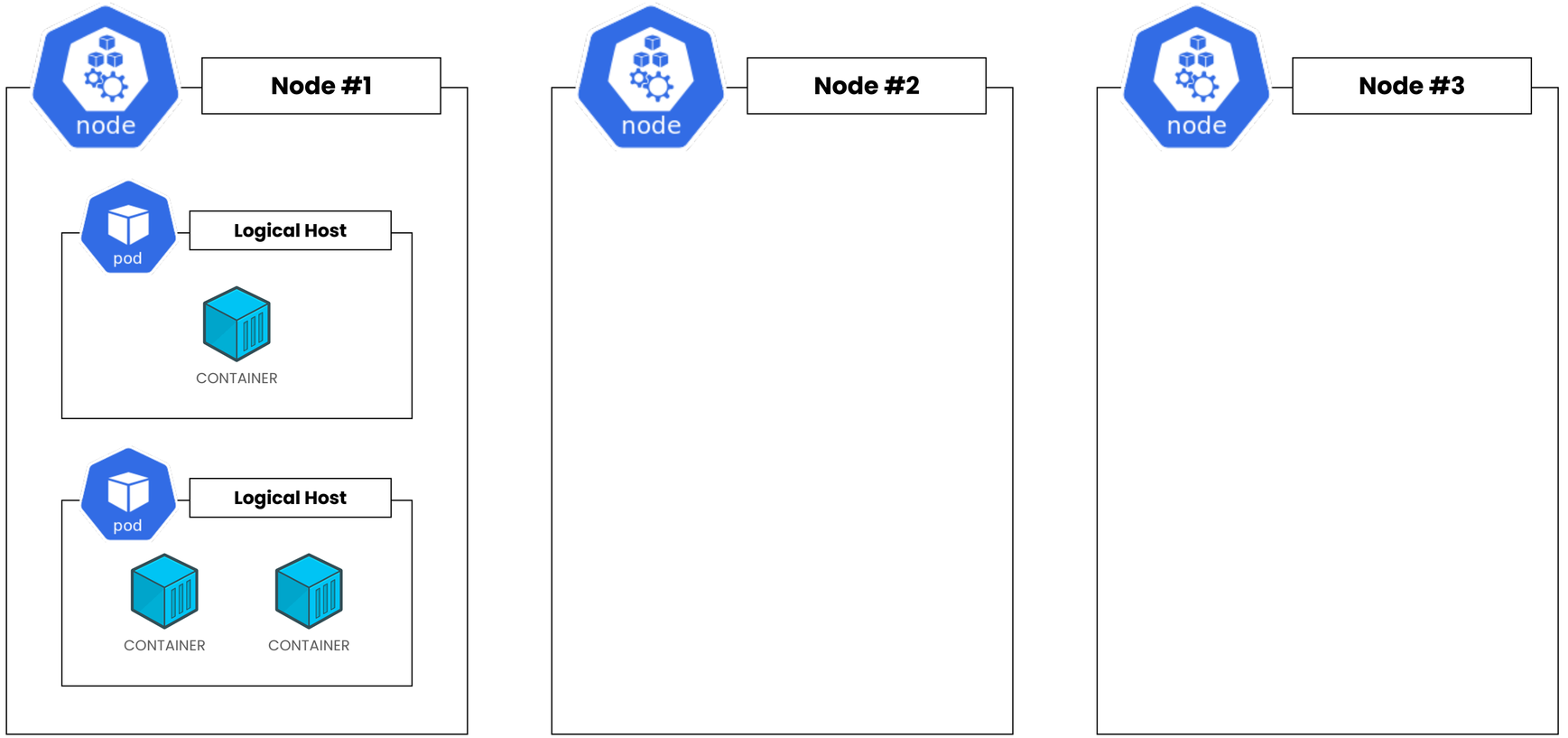
Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.



- Multi Pod in each node
- Doing the actual work

KUBERNETES ARCHITECTURE - NODE COMPONENT (WORKER NODE)

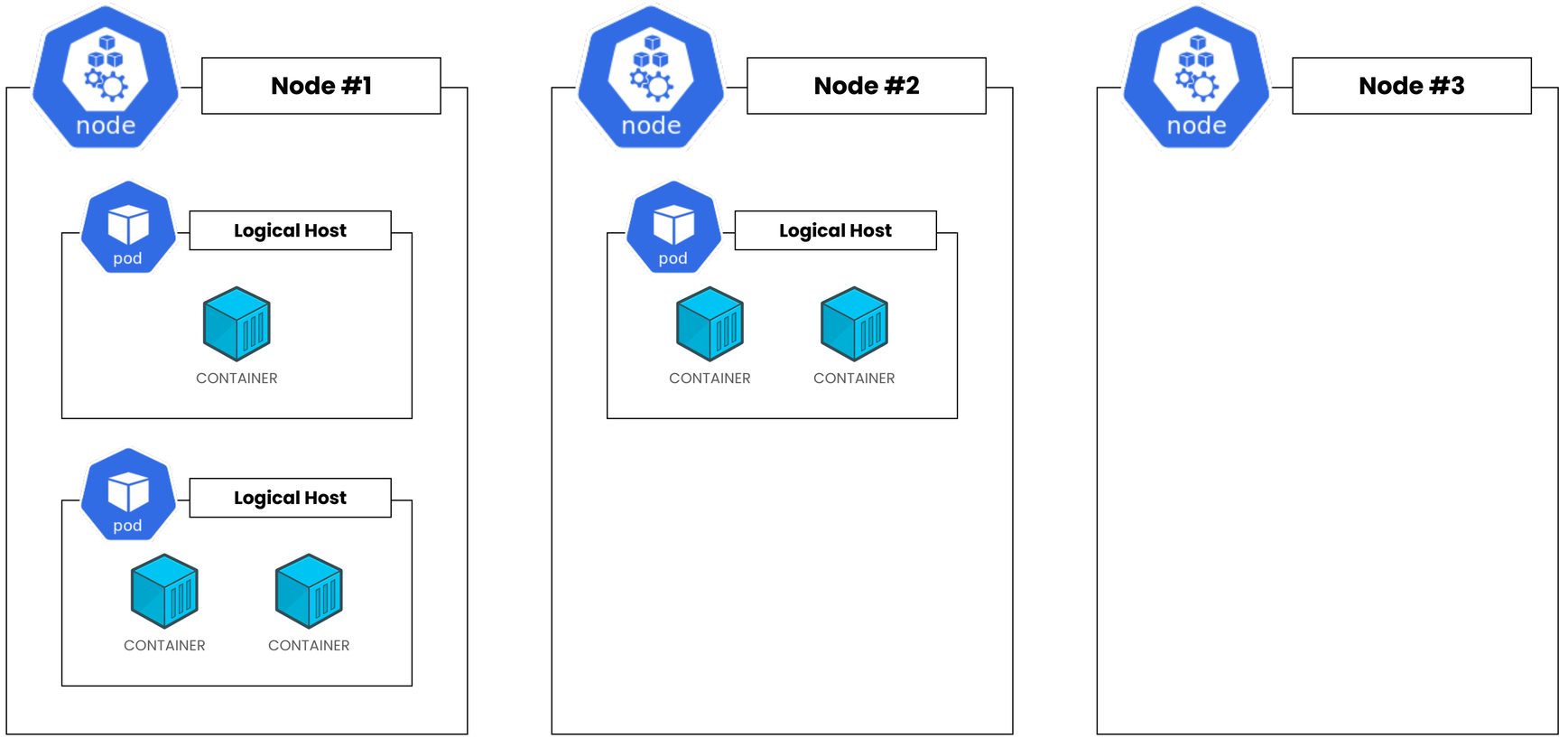
Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.



- Multi Pod in each node
- Doing the actual work

KUBERNETES ARCHITECTURE - NODE COMPONENT (WORKER NODE)

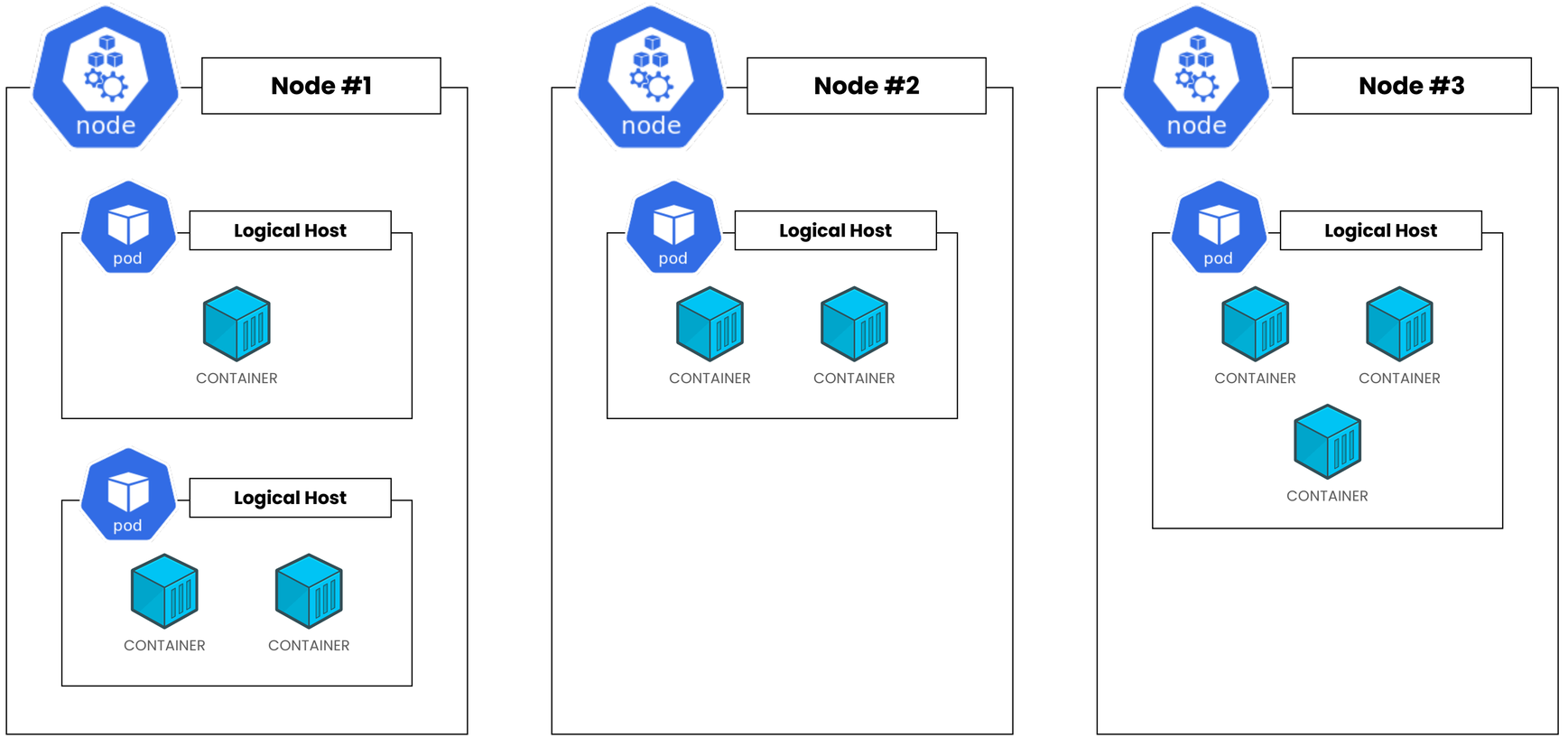
Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.



- Multi Pod in each node
- Doing the actual work

KUBERNETES ARCHITECTURE - NODE COMPONENT (WORKER NODE)

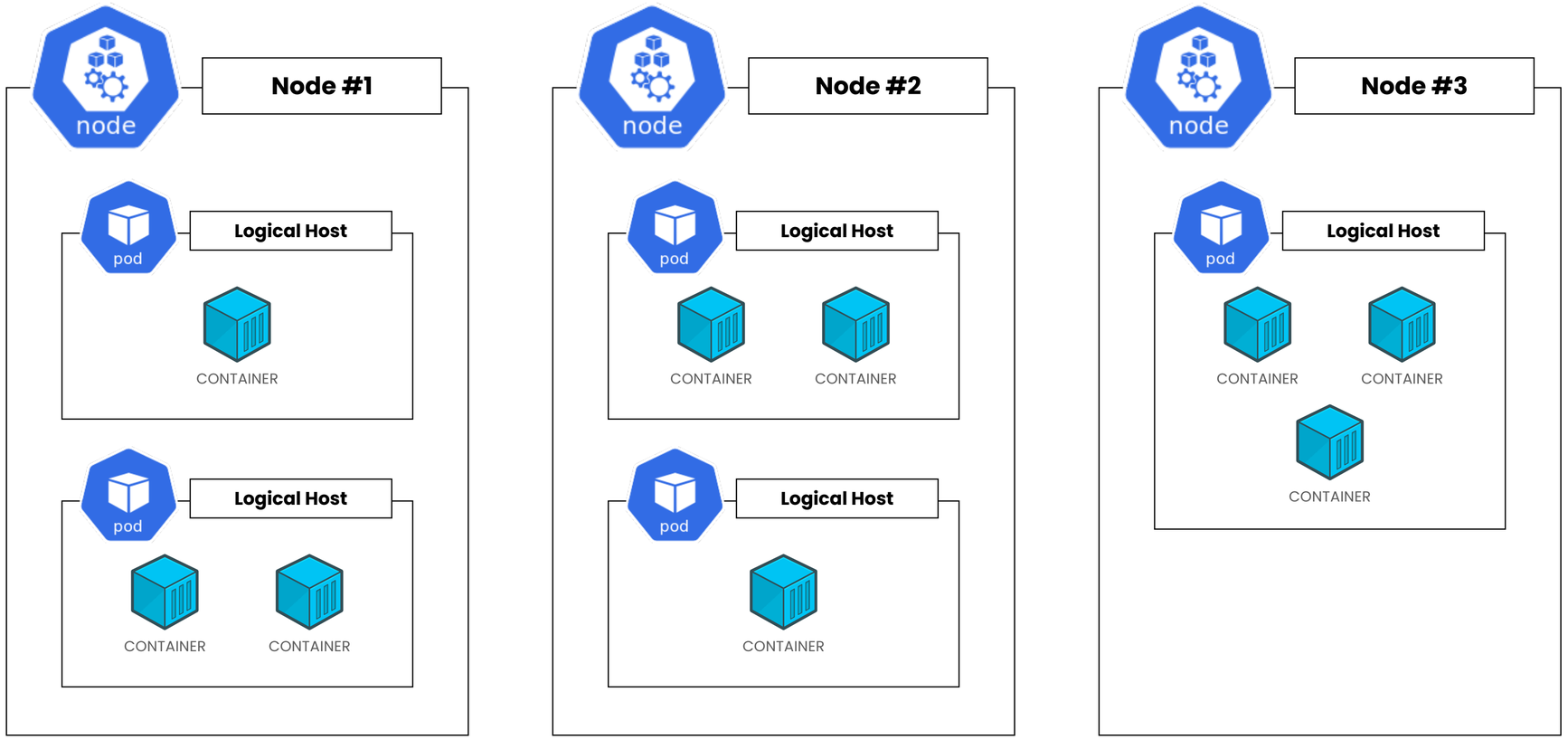
Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.



- Multi Pod in each node
- Doing the actual work

KUBERNETES ARCHITECTURE - NODE COMPONENT (WORKER NODE)

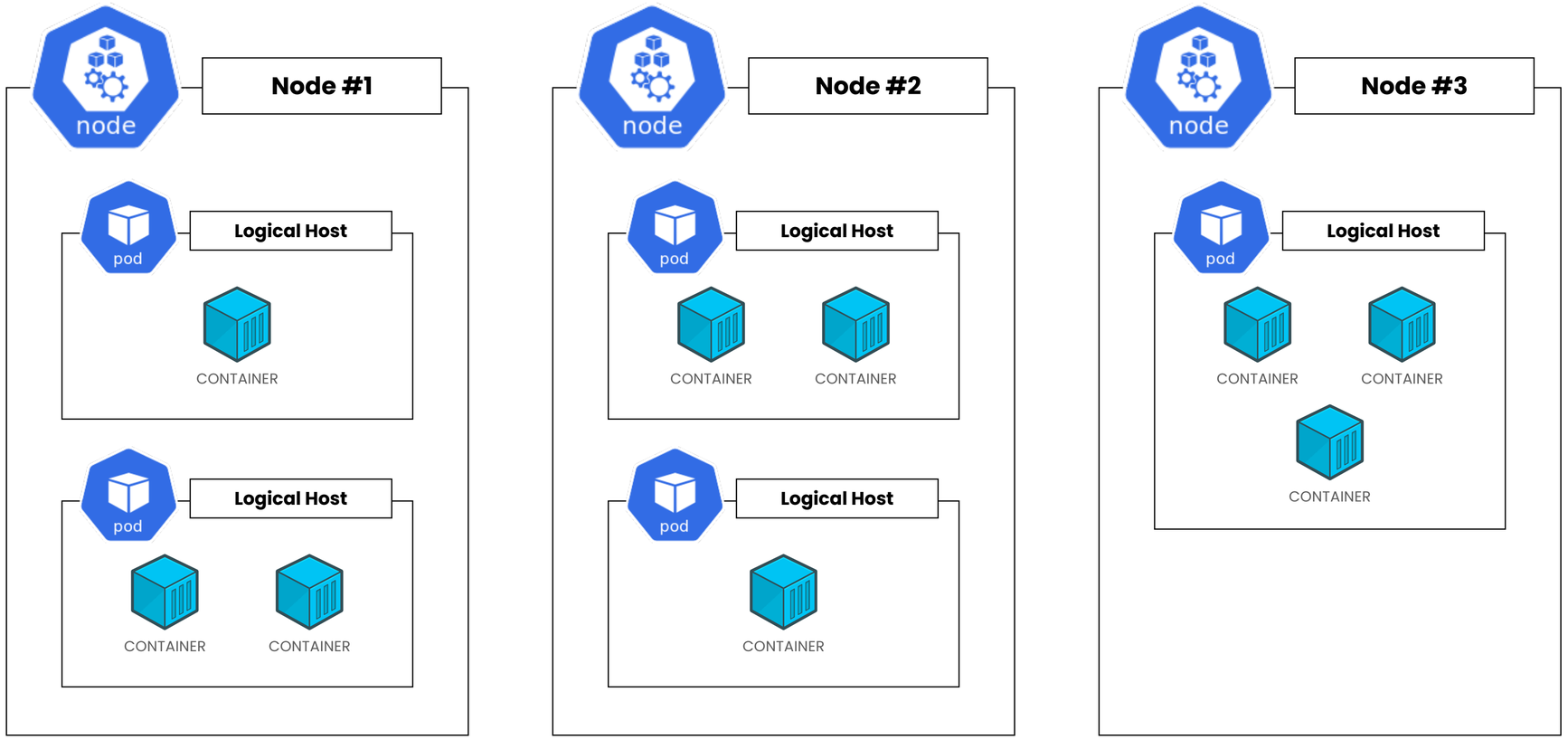
Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.



- Multi Pod in each node
- Doing the actual work

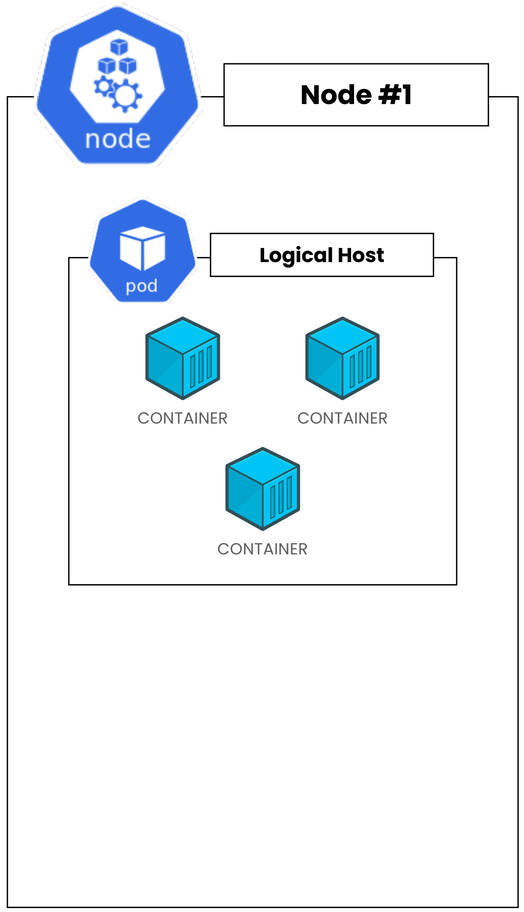
KUBERNETES ARCHITECTURE - NODE COMPONENT (WORKER NODE)

Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.



- Multi Pod in each node
- Doing the actual work
- 3 Processes installed in to Worker Node

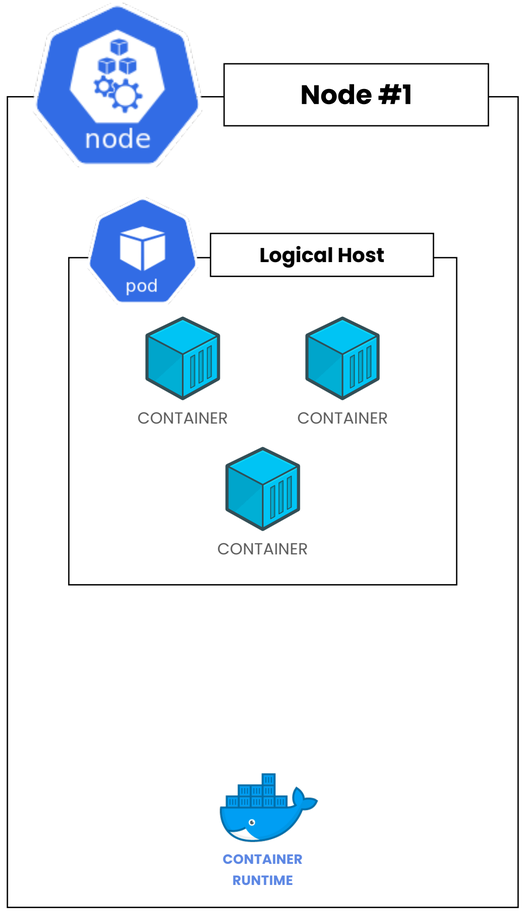
WORKER NODE - CONTAINER RUNTIME



The container runtime is the software that is responsible for running containers.

Kubernetes supports container runtimes such as containerd, CRI-O, and any other implementation of the Kubernetes CRI (Container Runtime Interface).

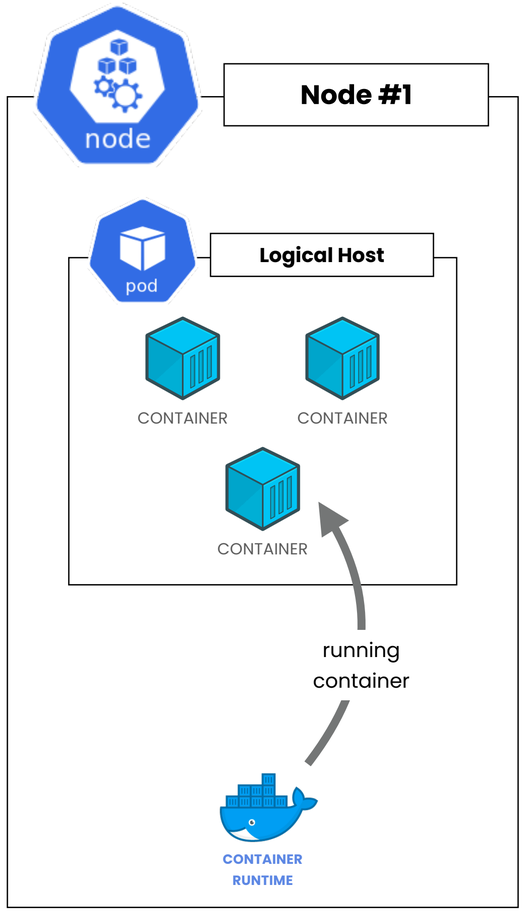
WORKER NODE - CONTAINER RUNTIME



The container runtime is the software that is responsible for running containers.

Kubernetes supports container runtimes such as containerd, CRI-O, and any other implementation of the Kubernetes CRI (Container Runtime Interface).

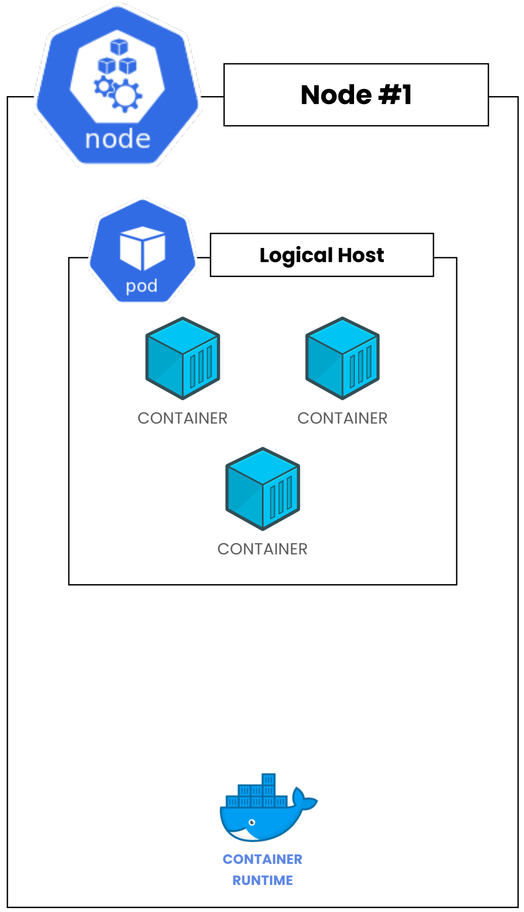
WORKER NODE - CONTAINER RUNTIME



The container runtime is the software that is responsible for running containers.

Kubernetes supports container runtimes such as containerd, CRI-O, and any other implementation of the Kubernetes CRI (Container Runtime Interface).

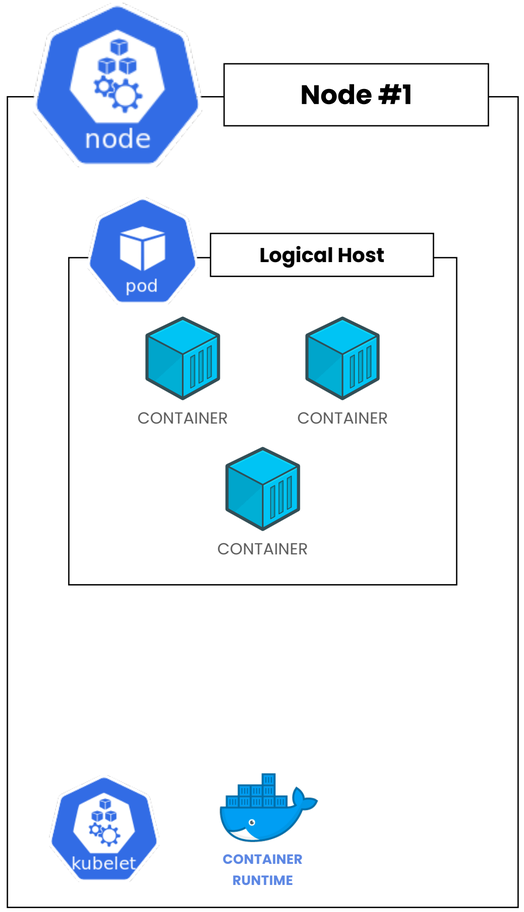
WORKER NODE - KUBELET



An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.

The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

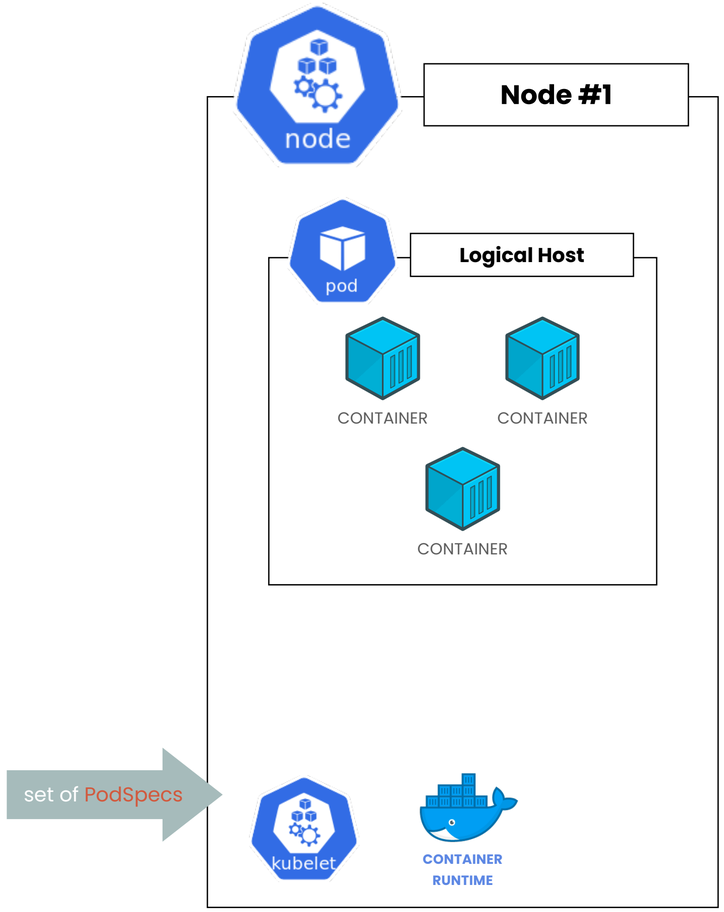
WORKER NODE - KUBELET



An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.

The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

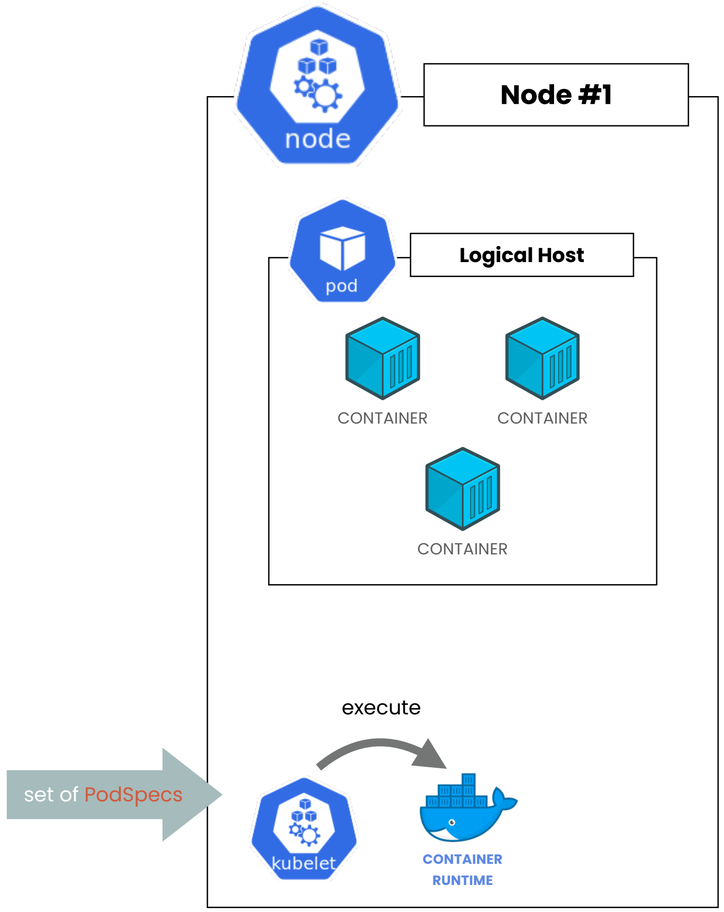
WORKER NODE - KUBELET



An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.

The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

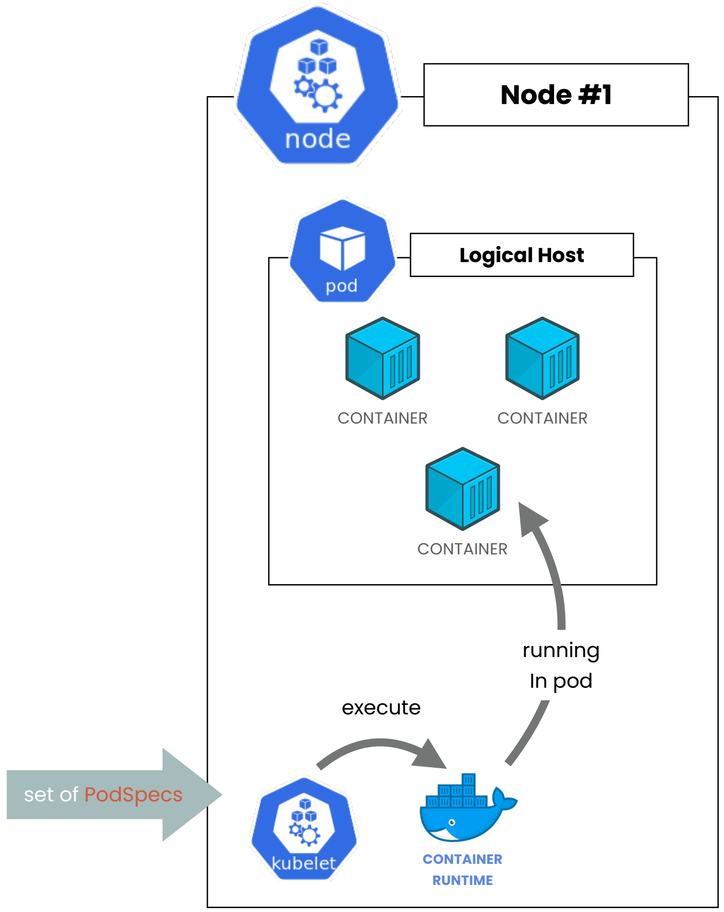
WORKER NODE - KUBELET



An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.

The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

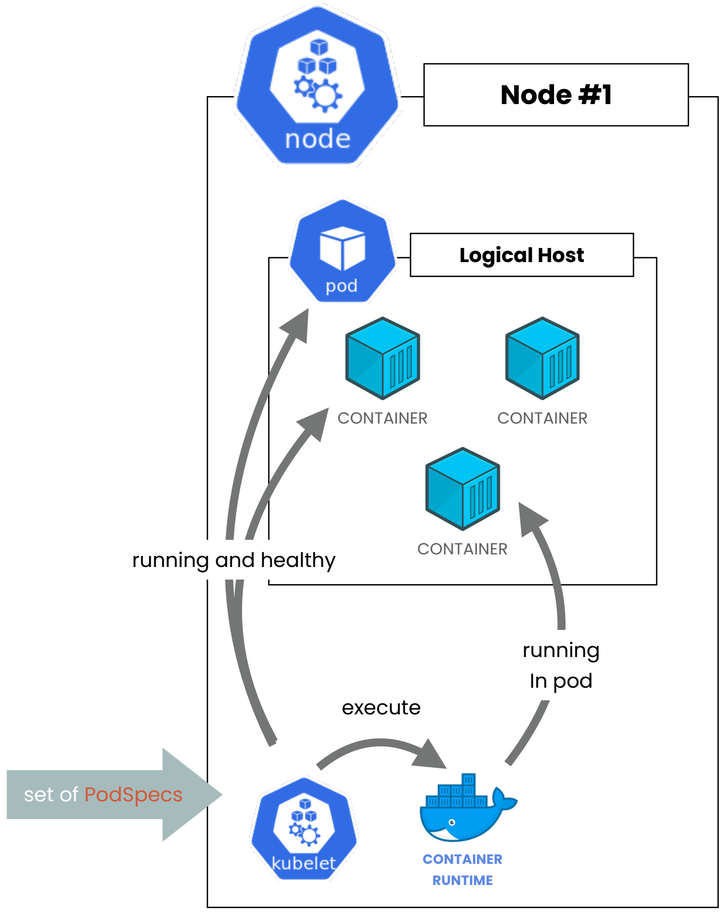
WORKER NODE - KUBELET



An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.

The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

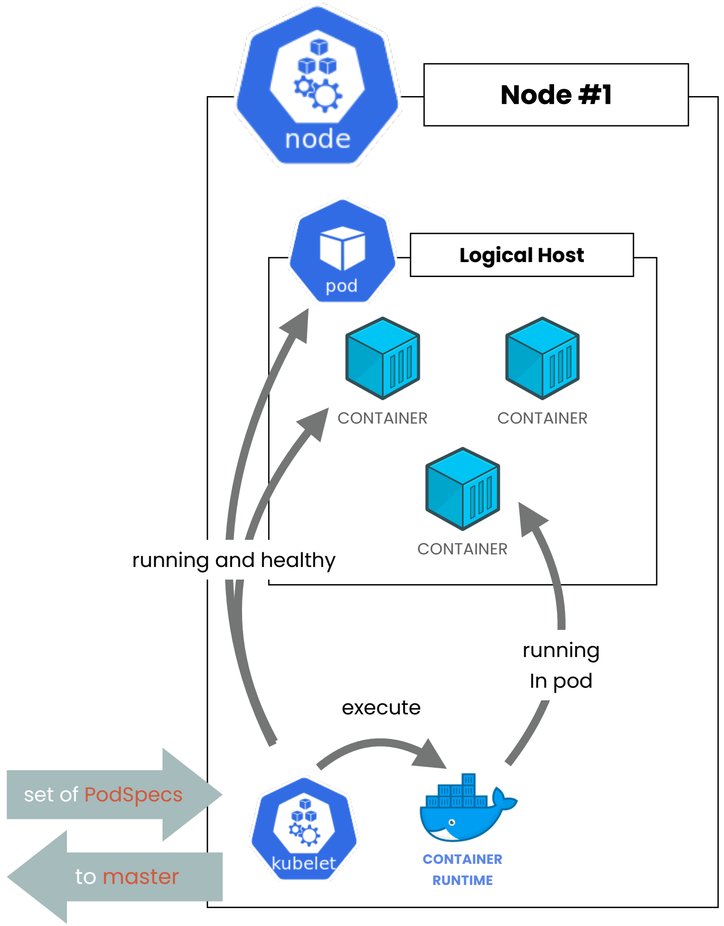
WORKER NODE - KUBELET



An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.

The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

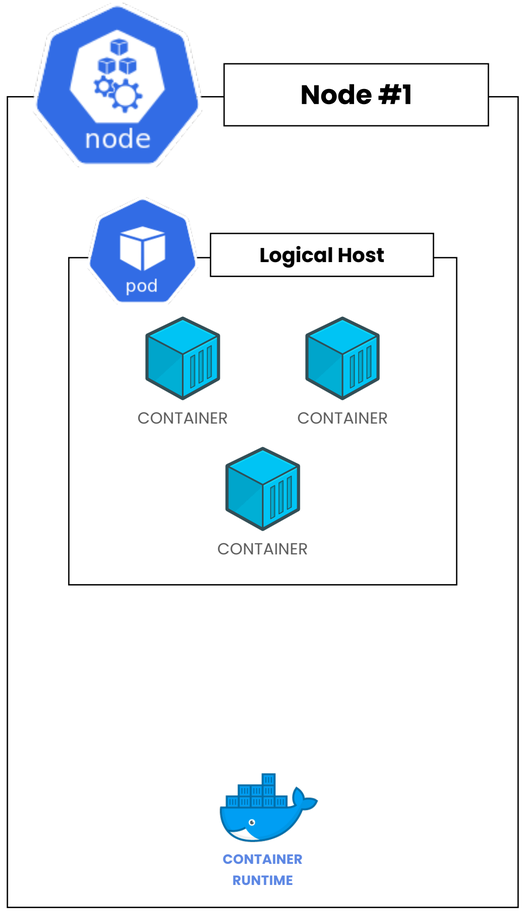
WORKER NODE - KUBELET



An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.

The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

WORKER NODE - KUBE-PROXY

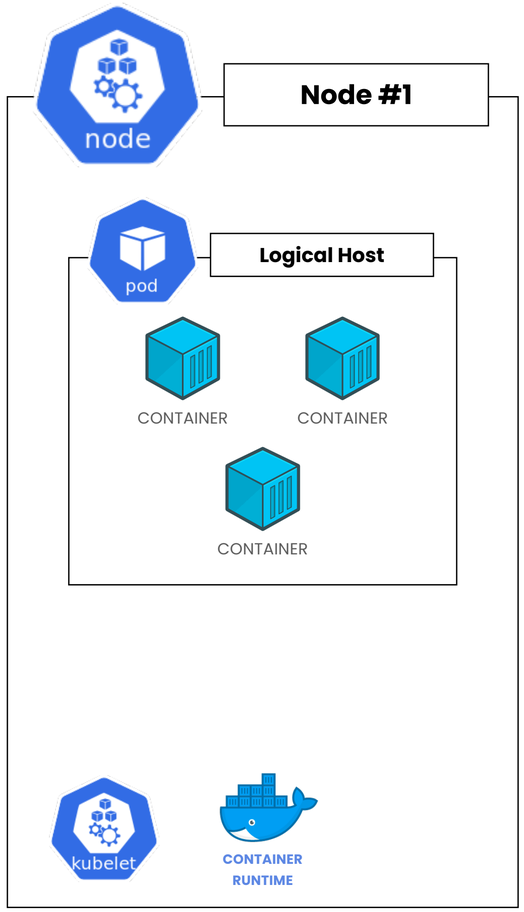


kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept.

kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.

kube-proxy uses the operating system packet filtering layer if there is one and it's available. Otherwise, kube-proxy forwards the traffic itself.

WORKER NODE - KUBE-PROXY

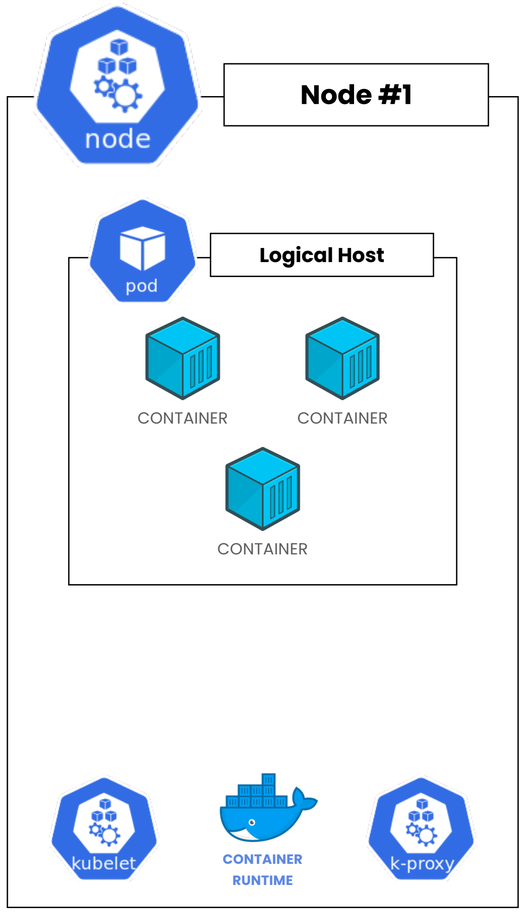


kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept.

kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.

kube-proxy uses the operating system packet filtering layer if there is one and it's available. Otherwise, kube-proxy forwards the traffic itself.

WORKER NODE - KUBE-PROXY

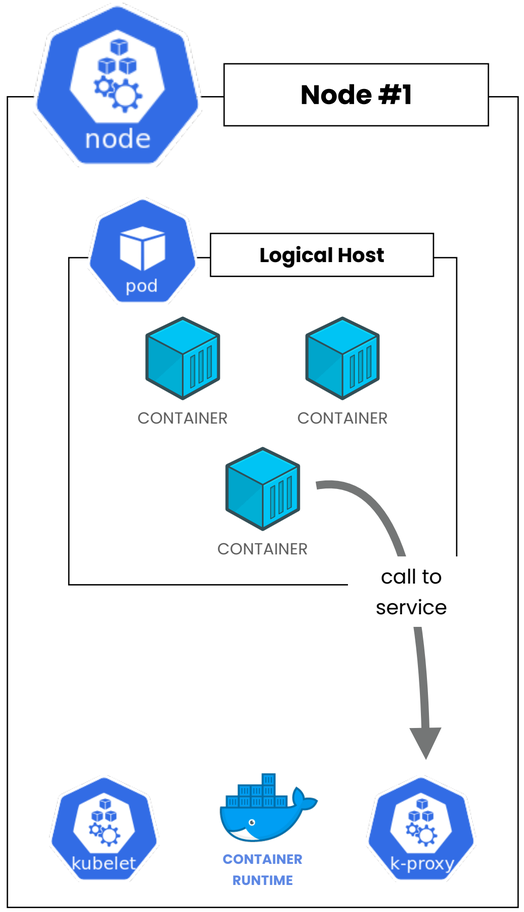


kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept.

kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.

kube-proxy uses the operating system packet filtering layer if there is one and it's available. Otherwise, kube-proxy forwards the traffic itself.

WORKER NODE - KUBE-PROXY

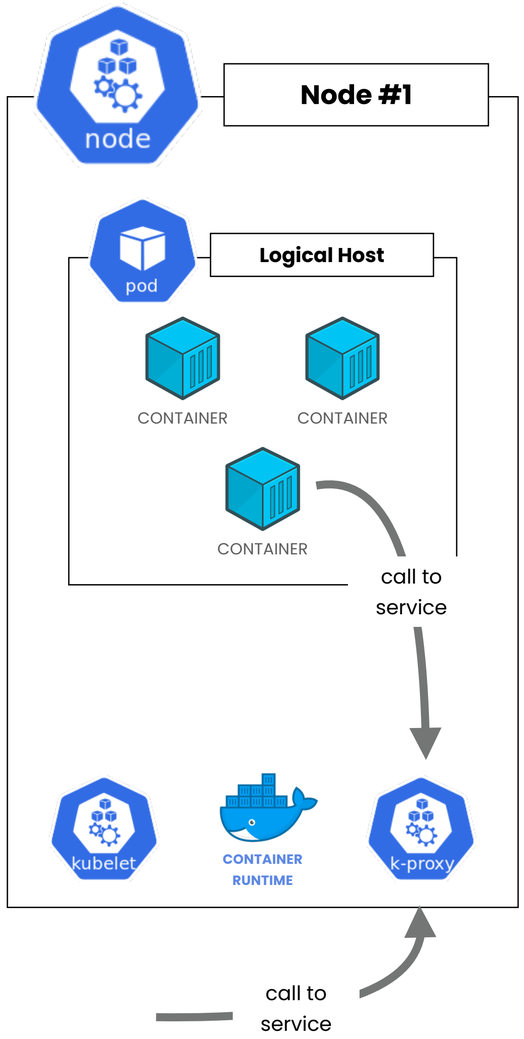


kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept.

kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.

kube-proxy uses the operating system packet filtering layer if there is one and it's available. Otherwise, kube-proxy forwards the traffic itself.

WORKER NODE - KUBE-PROXY

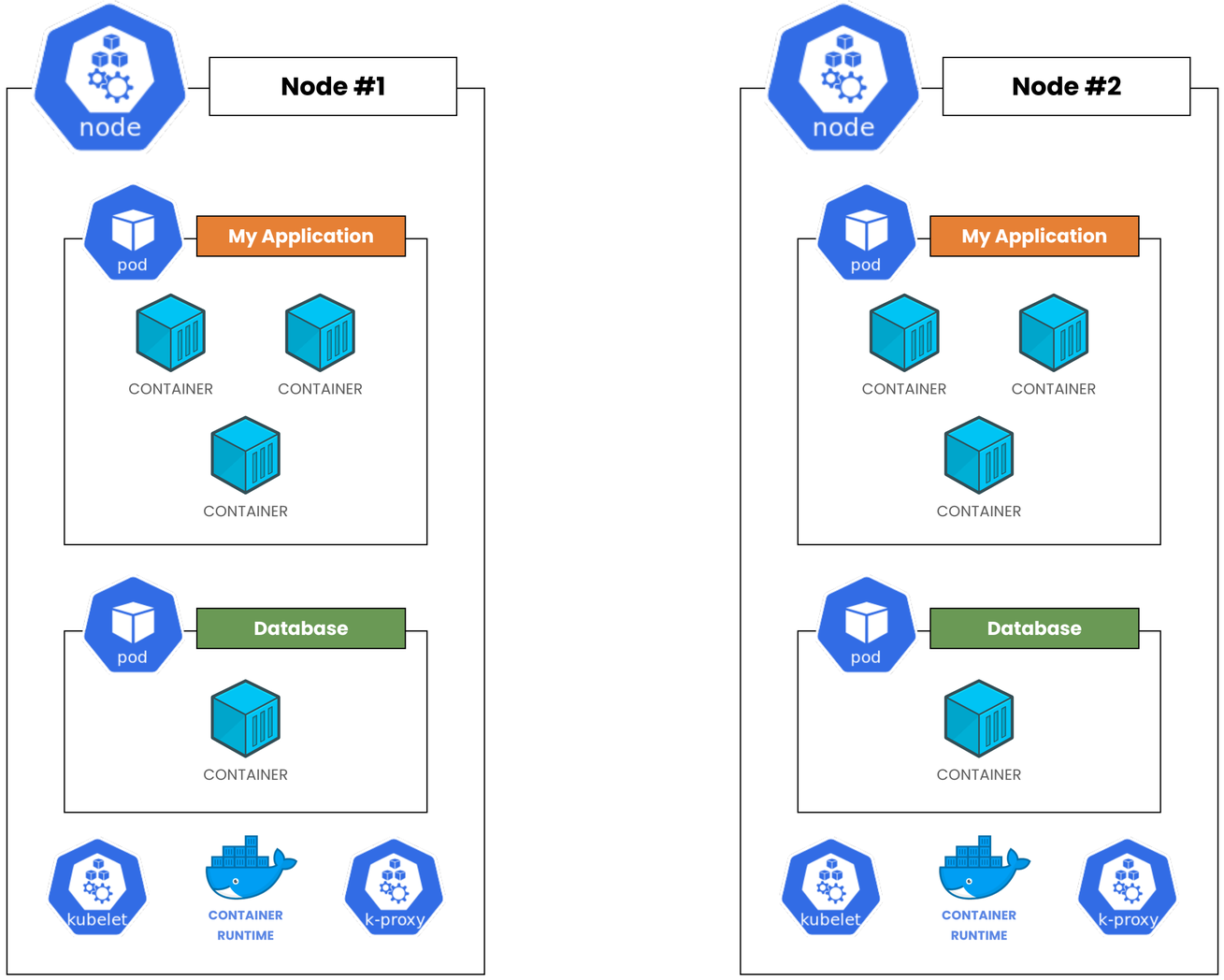


kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept.

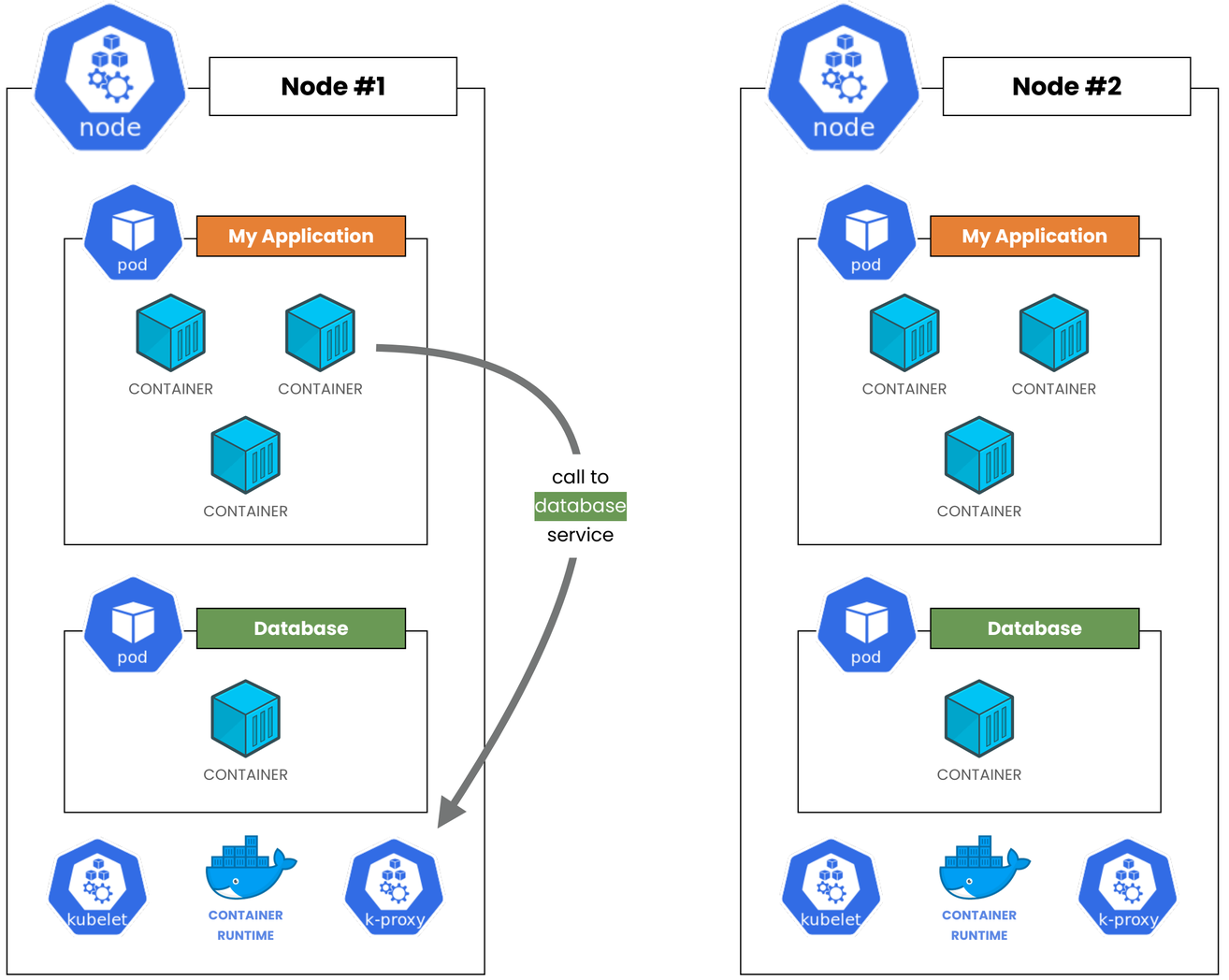
kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.

kube-proxy uses the operating system packet filtering layer if there is one and it's available. Otherwise, kube-proxy forwards the traffic itself.

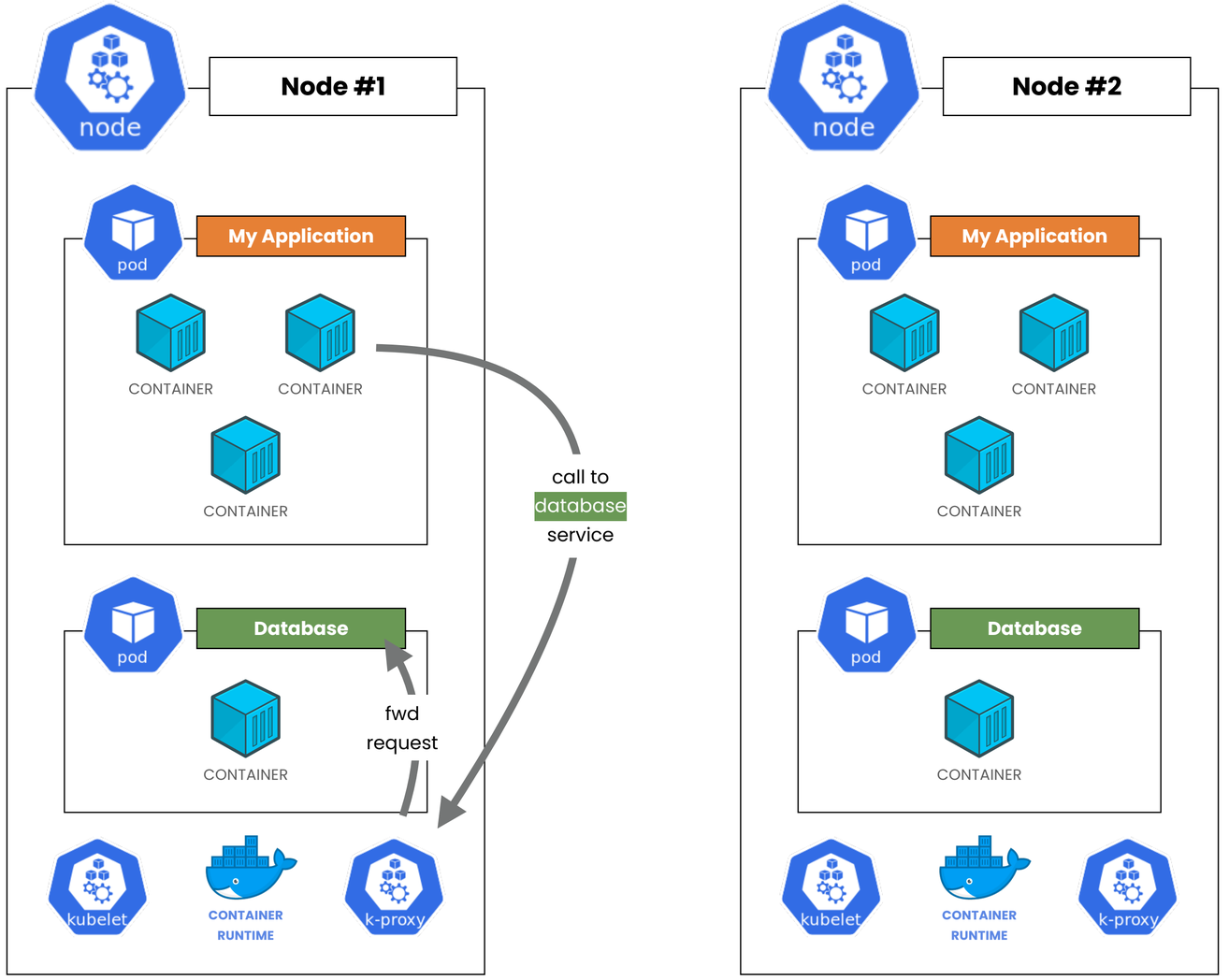
WORKER NODE - KUBE-PROXY



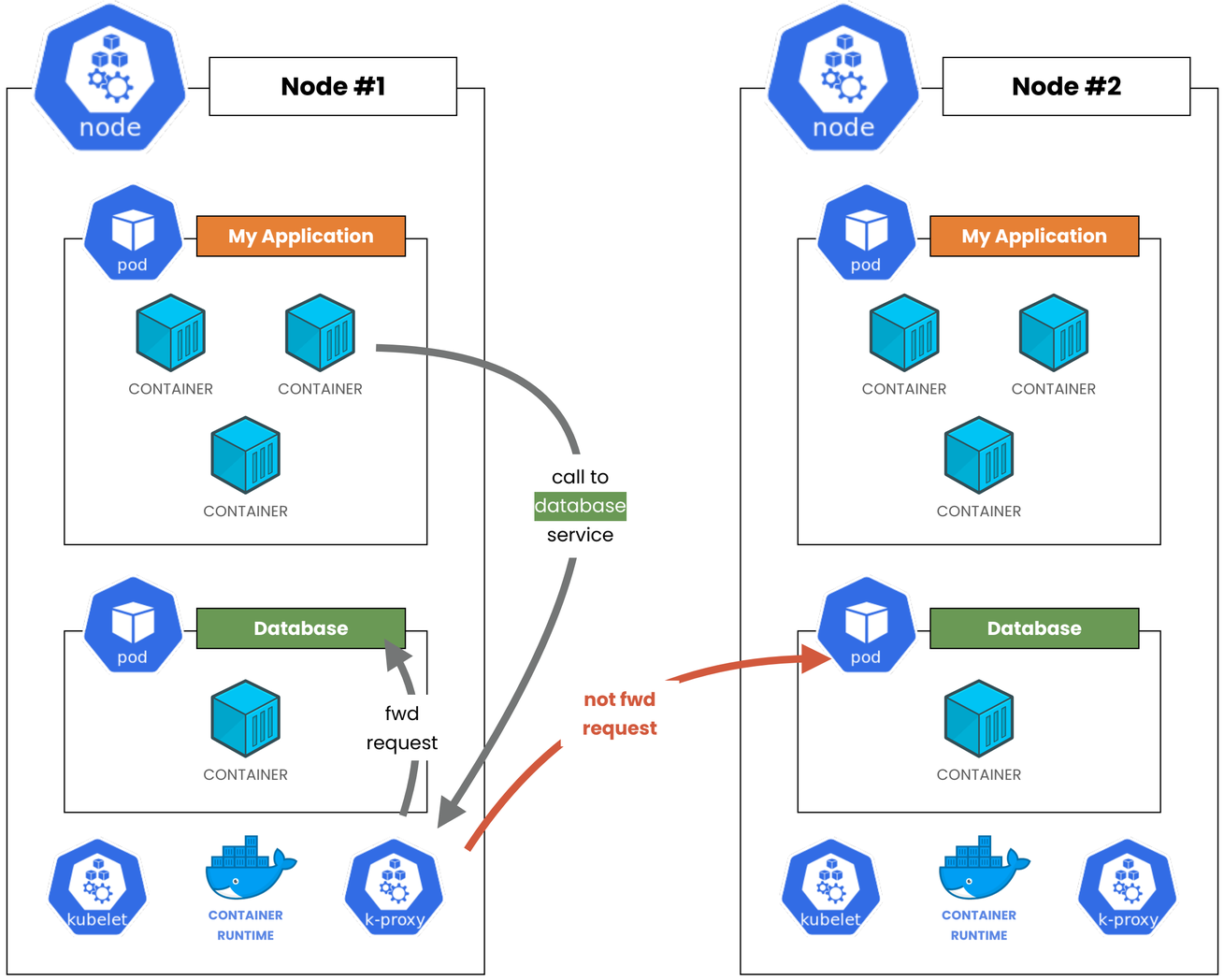
WORKER NODE - KUBE-PROXY



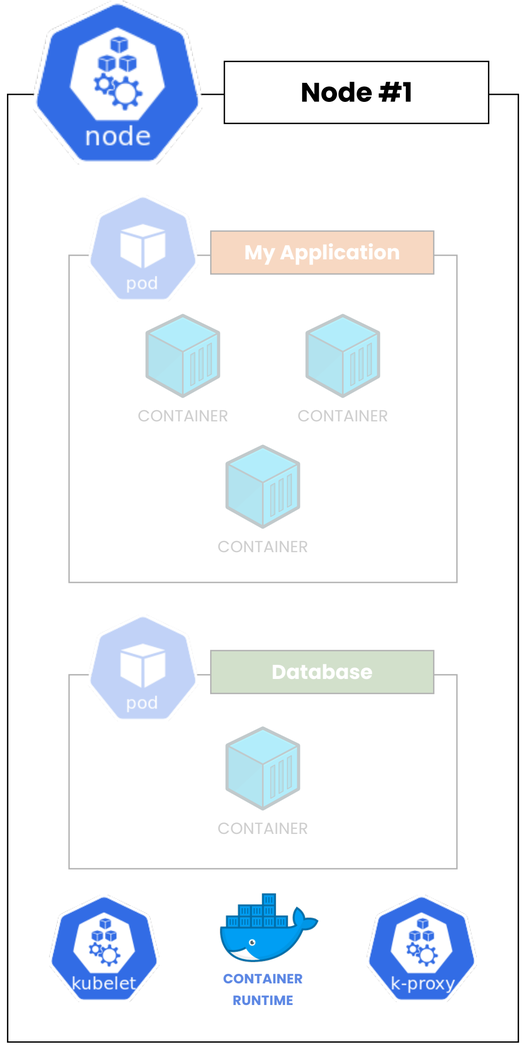
WORKER NODE - KUBE-PROXY



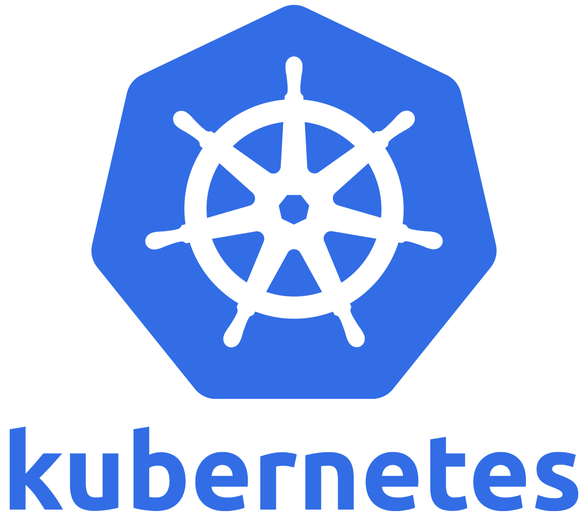
WORKER NODE - KUBE-PROXY



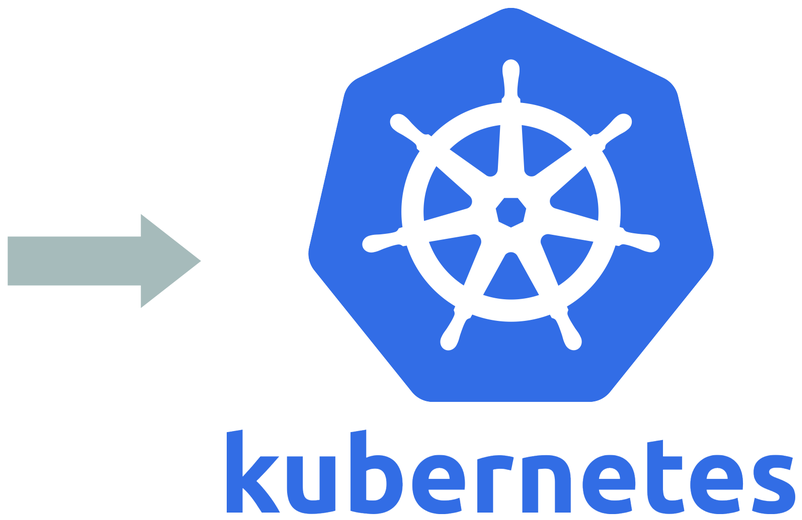
WORKER NODE - COMPONENT



HOW CAN WE INTERACT WITH KUBERNETES CLUSTER

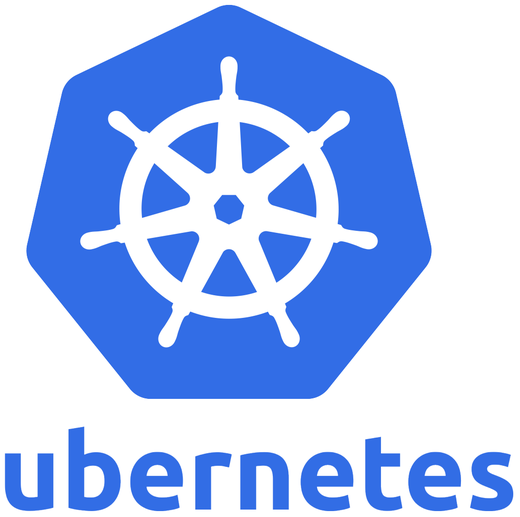


HOW CAN WE INTERACT WITH KUBERNETES CLUSTER



HOW CAN WE INTERACT WITH KUBERNETES CLUSTER

- Schedule Pod
- Monitor
- Re-Schedule/Restart Pod
- Join a new Node



CONTROL PLANE COMPONENT

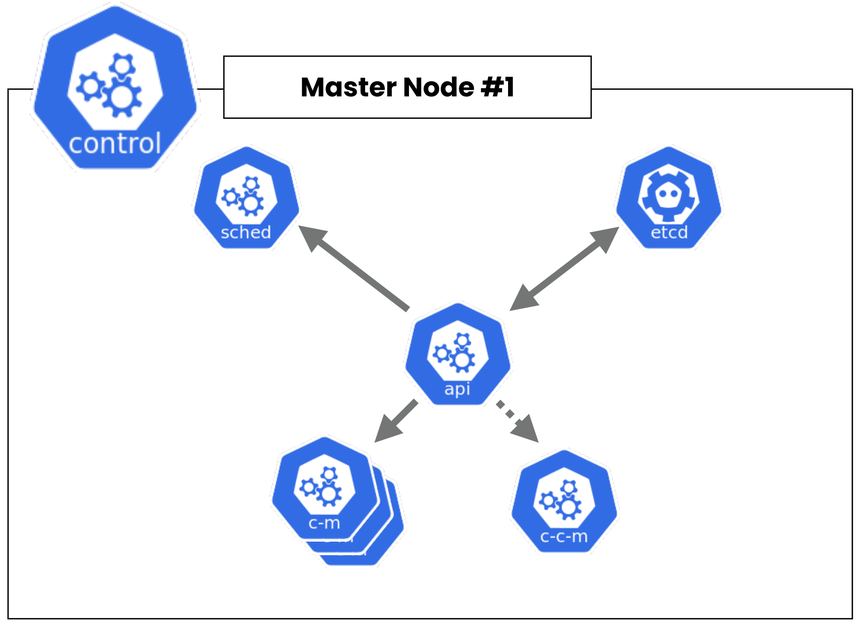
–

MASTER NODES

KUBERNETES ARCHITECTURE - CONTROL PLANE COMPONENT (MASTER NODES)

The control plane's components make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment's replicas field is unsatisfied).

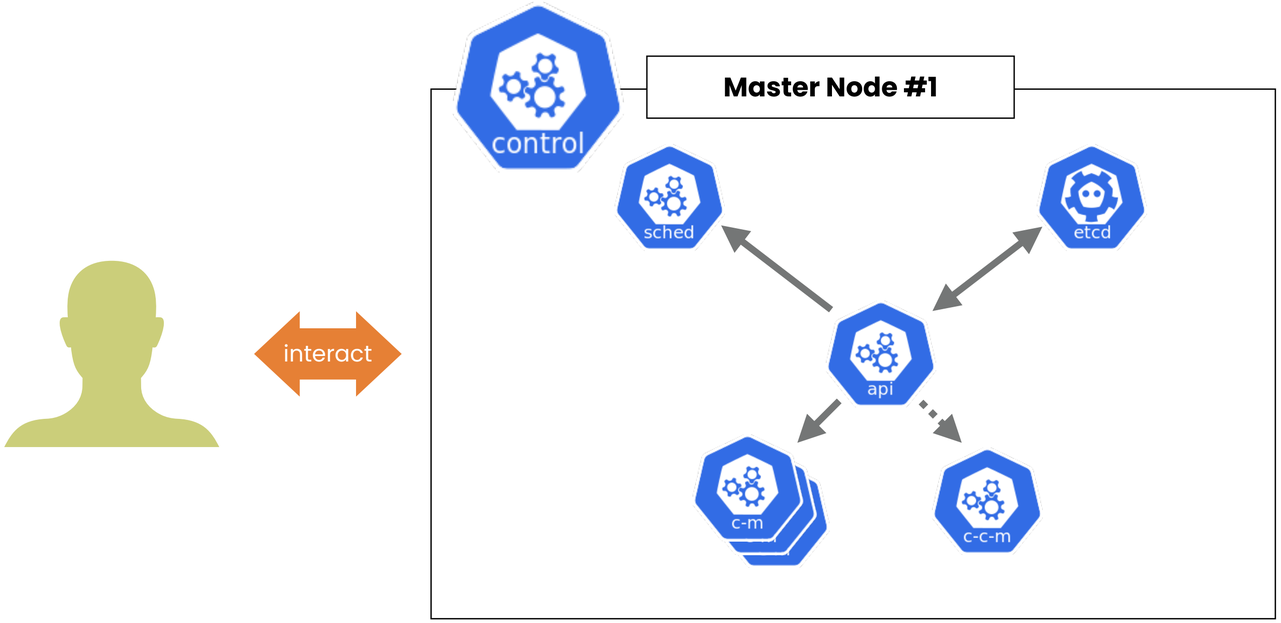
Control plane components can be run on any machine in the cluster. However, for simplicity, set up scripts typically start all control plane components on the same machine, and do not run user containers on this machine. See Creating Highly Available clusters with kubeadm for an example control plane setup that runs across multiple VMs.



KUBERNETES ARCHITECTURE - CONTROL PLANE COMPONENT (MASTER NODES)

The control plane's components make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment's replicas field is unsatisfied).

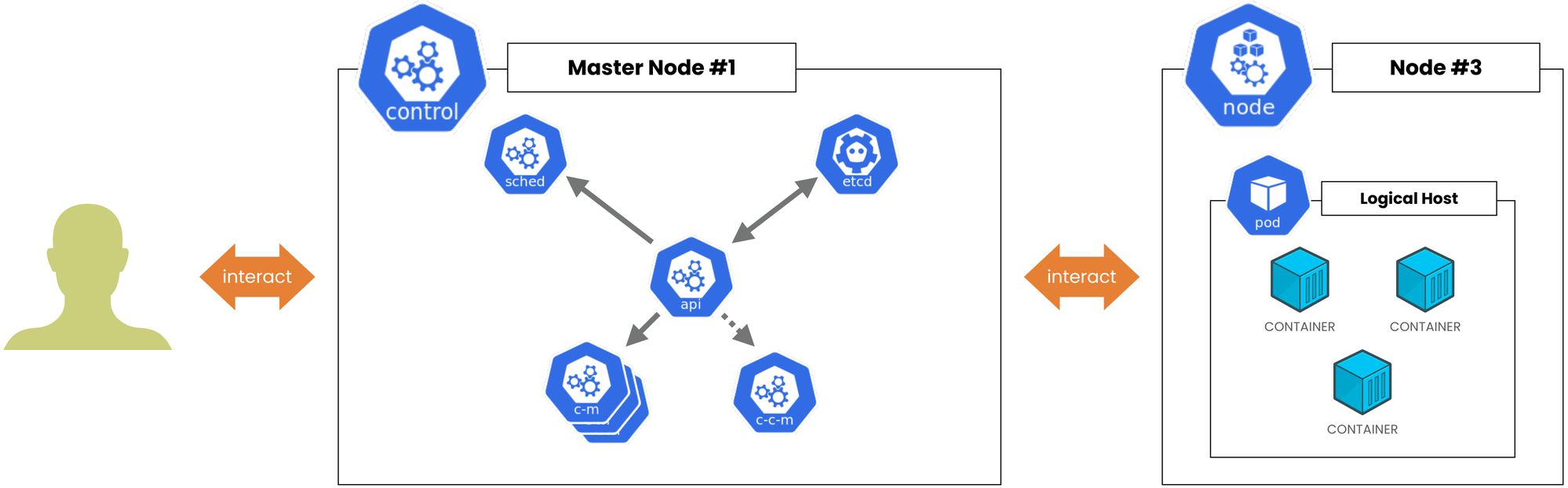
Control plane components can be run on any machine in the cluster. However, for simplicity, set up scripts typically start all control plane components on the same machine, and do not run user containers on this machine. See Creating Highly Available clusters with kubeadm for an example control plane setup that runs across multiple VMs.



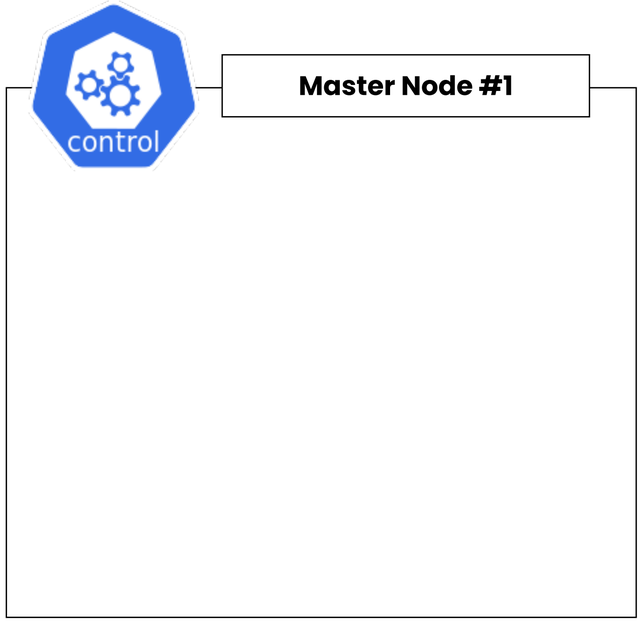
KUBERNETES ARCHITECTURE - CONTROL PLANE COMPONENT (MASTER NODES)

The control plane's components make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment's replicas field is unsatisfied).

Control plane components can be run on any machine in the cluster. However, for simplicity, set up scripts typically start all control plane components on the same machine, and do not run user containers on this machine. See Creating Highly Available clusters with kubeadm for an example control plane setup that runs across multiple VMs.



MASTER NODE - KUBE-APISERVER

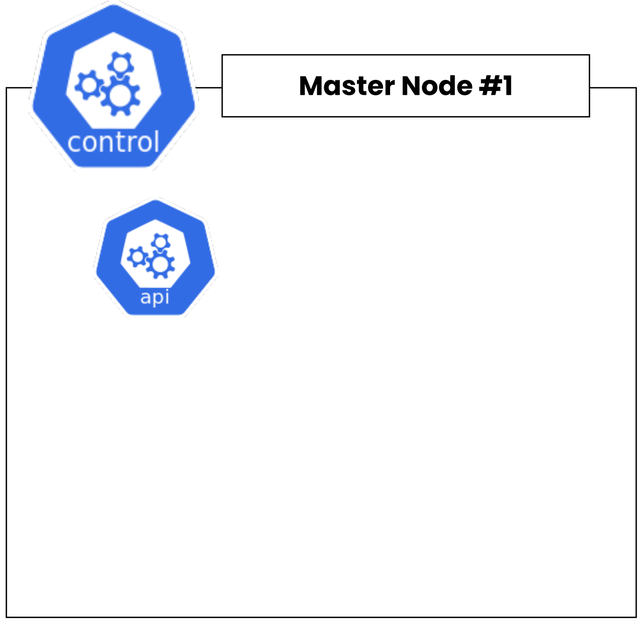


The API server is the front end (Cluster Gateway) for the Kubernetes control plane.

The main implementation of a Kubernetes API server is kube-apiserver.

kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of kube-apiserver and balance traffic between those instances.

MASTER NODE - KUBE-APISERVER

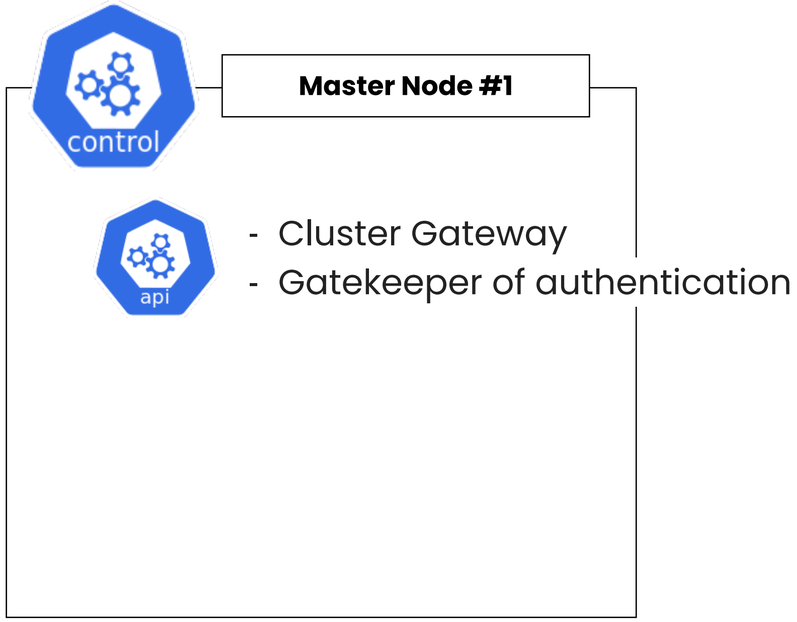


The API server is the front end (Cluster Gateway) for the Kubernetes control plane.

The main implementation of a Kubernetes API server is kube-apiserver.

kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of kube-apiserver and balance traffic between those instances.

MASTER NODE - KUBE-APISERVER

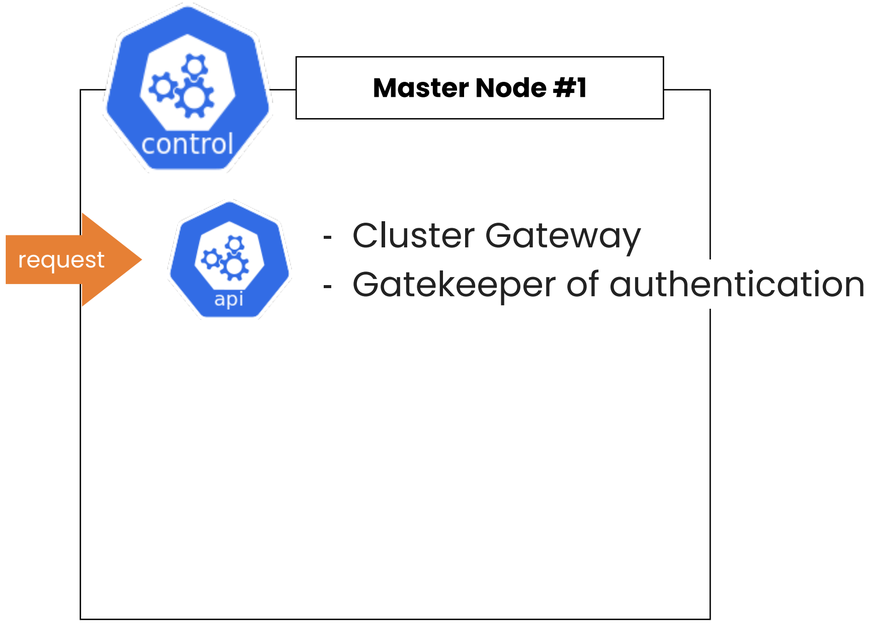


The API server is the front end (Cluster Gateway) for the Kubernetes control plane.

The main implementation of a Kubernetes API server is kube-apiserver.

kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of kube-apiserver and balance traffic between those instances.

MASTER NODE - KUBE-APISERVER

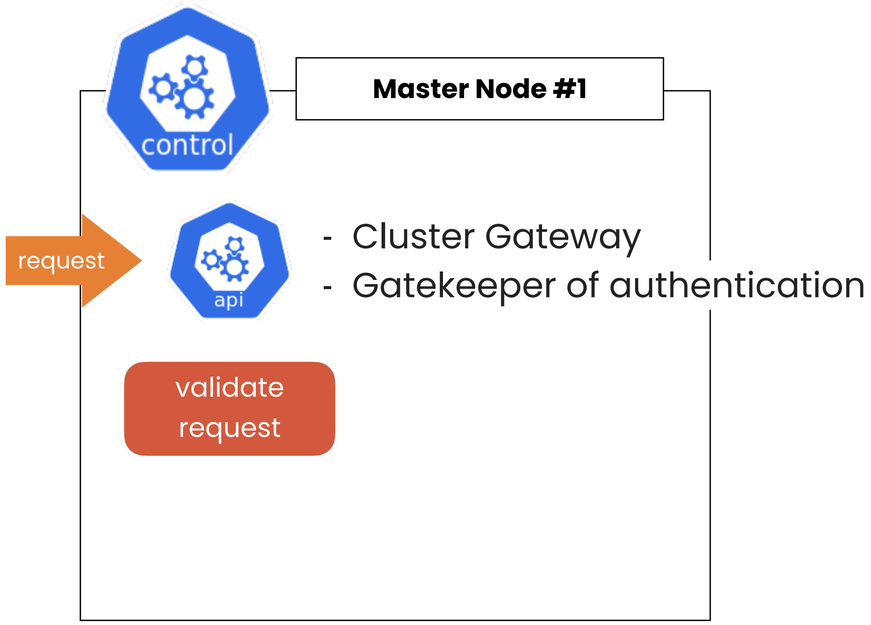


The API server is the front end (Cluster Gateway) for the Kubernetes control plane.

The main implementation of a Kubernetes API server is kube-apiserver.

kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of kube-apiserver and balance traffic between those instances.

MASTER NODE - KUBE-APISERVER

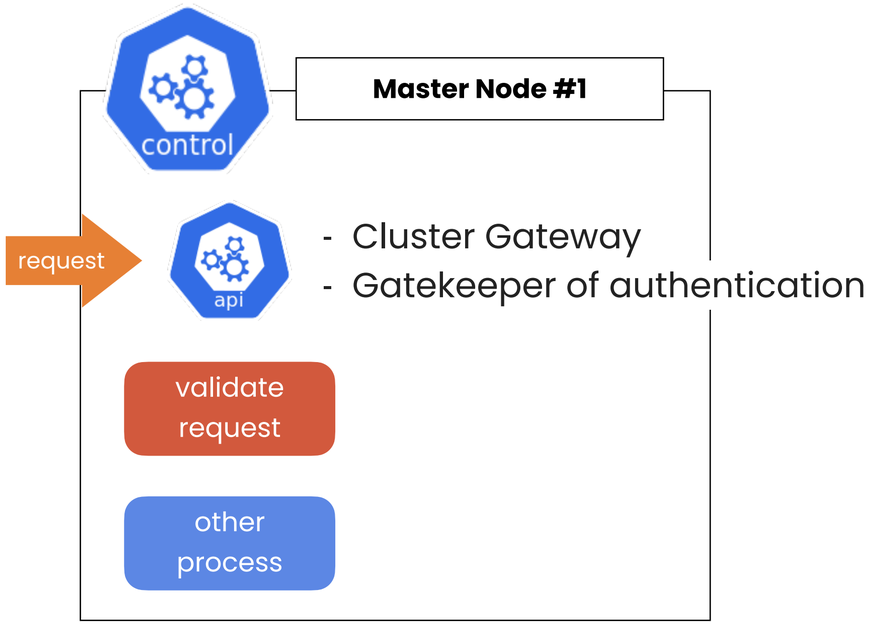


The API server is the front end (Cluster Gateway) for the Kubernetes control plane.

The main implementation of a Kubernetes API server is kube-apiserver.

kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of kube-apiserver and balance traffic between those instances.

MASTER NODE - KUBE-APISERVER

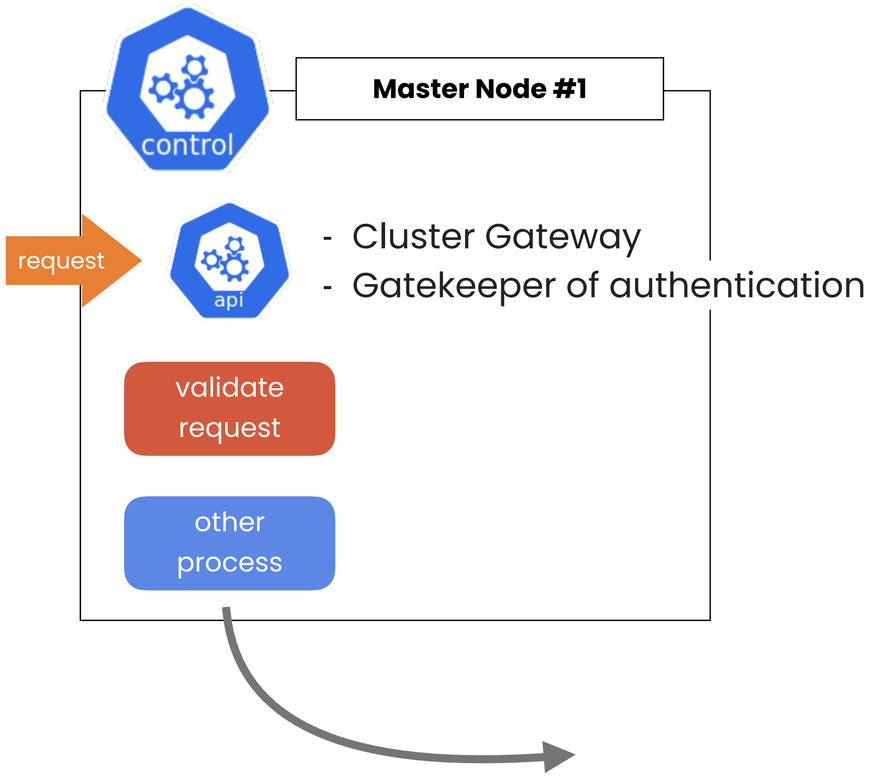


The API server is the front end (Cluster Gateway) for the Kubernetes control plane.

The main implementation of a Kubernetes API server is kube-apiserver.

kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of kube-apiserver and balance traffic between those instances.

MASTER NODE - KUBE-APISERVER

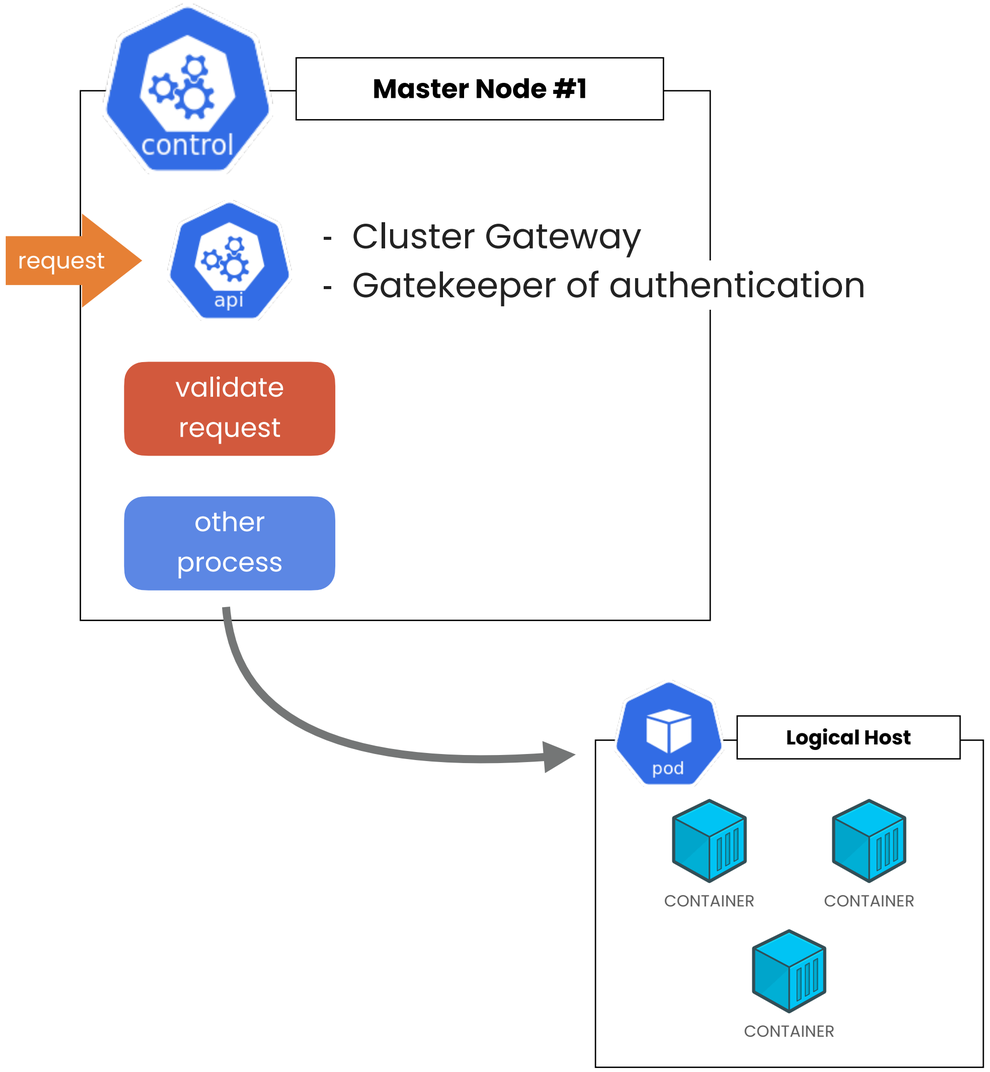


The API server is the front end (Cluster Gateway) for the Kubernetes control plane.

The main implementation of a Kubernetes API server is kube-apiserver.

kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of kube-apiserver and balance traffic between those instances.

MASTER NODE - KUBE-APISERVER

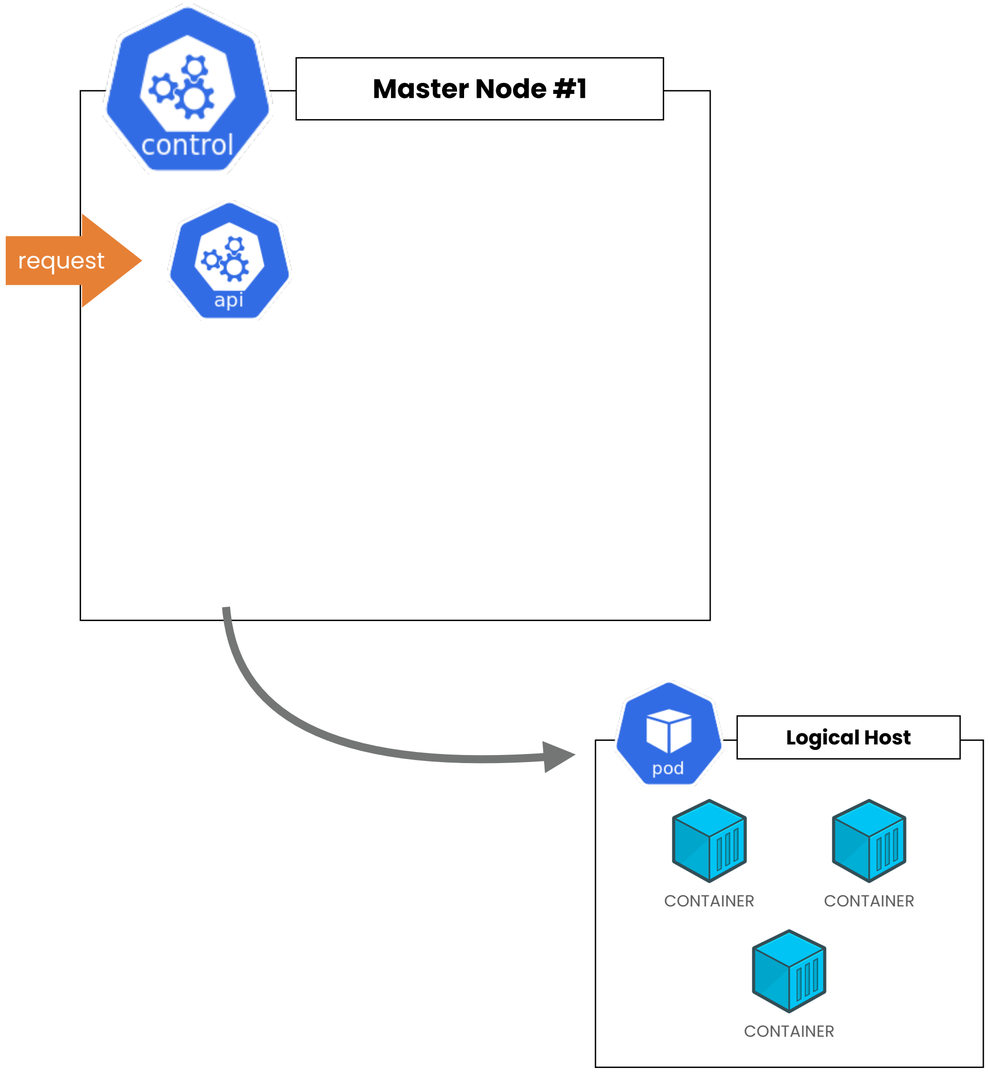


The API server is the front end (Cluster Gateway) for the Kubernetes control plane.

The main implementation of a Kubernetes API server is kube-apiserver.

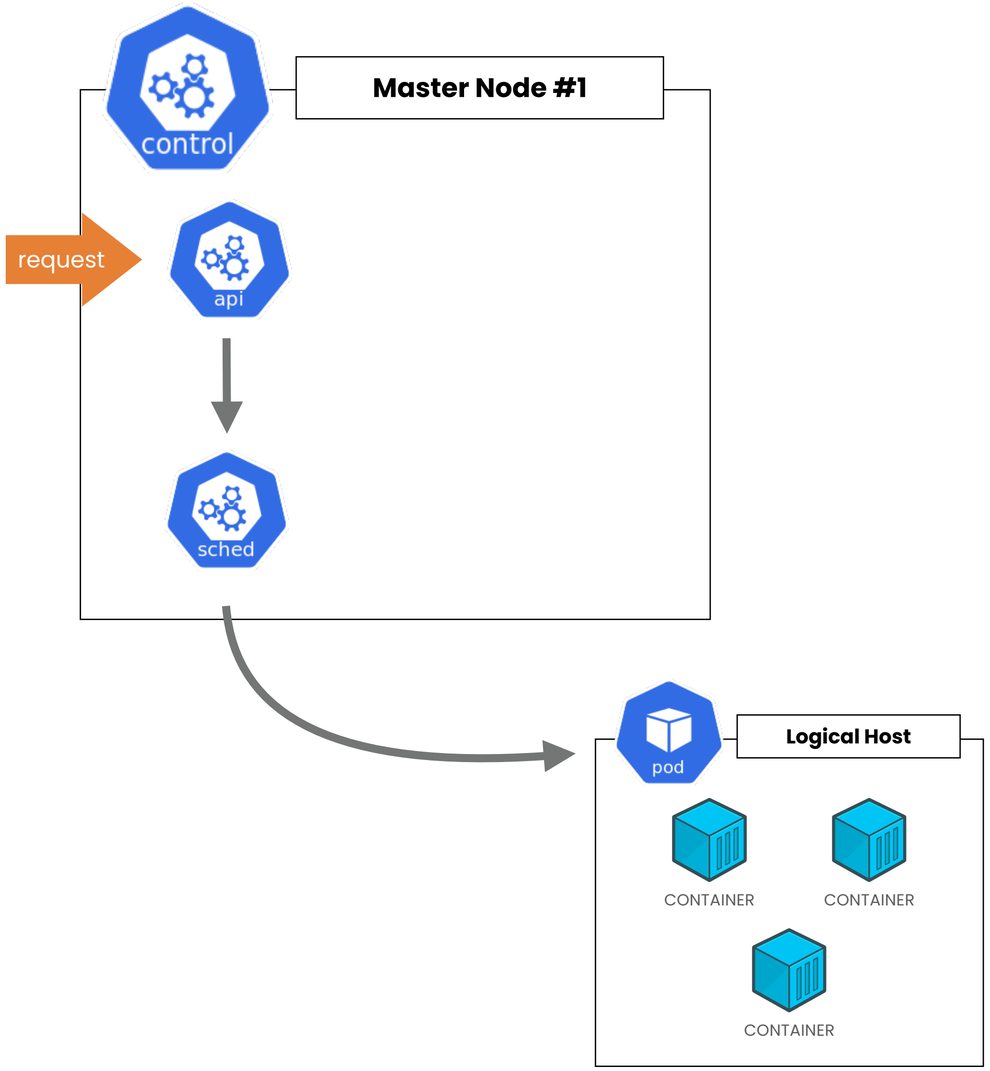
kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of kube-apiserver and balance traffic between those instances.

MASTER NODE - KUBE-SCHEDULER



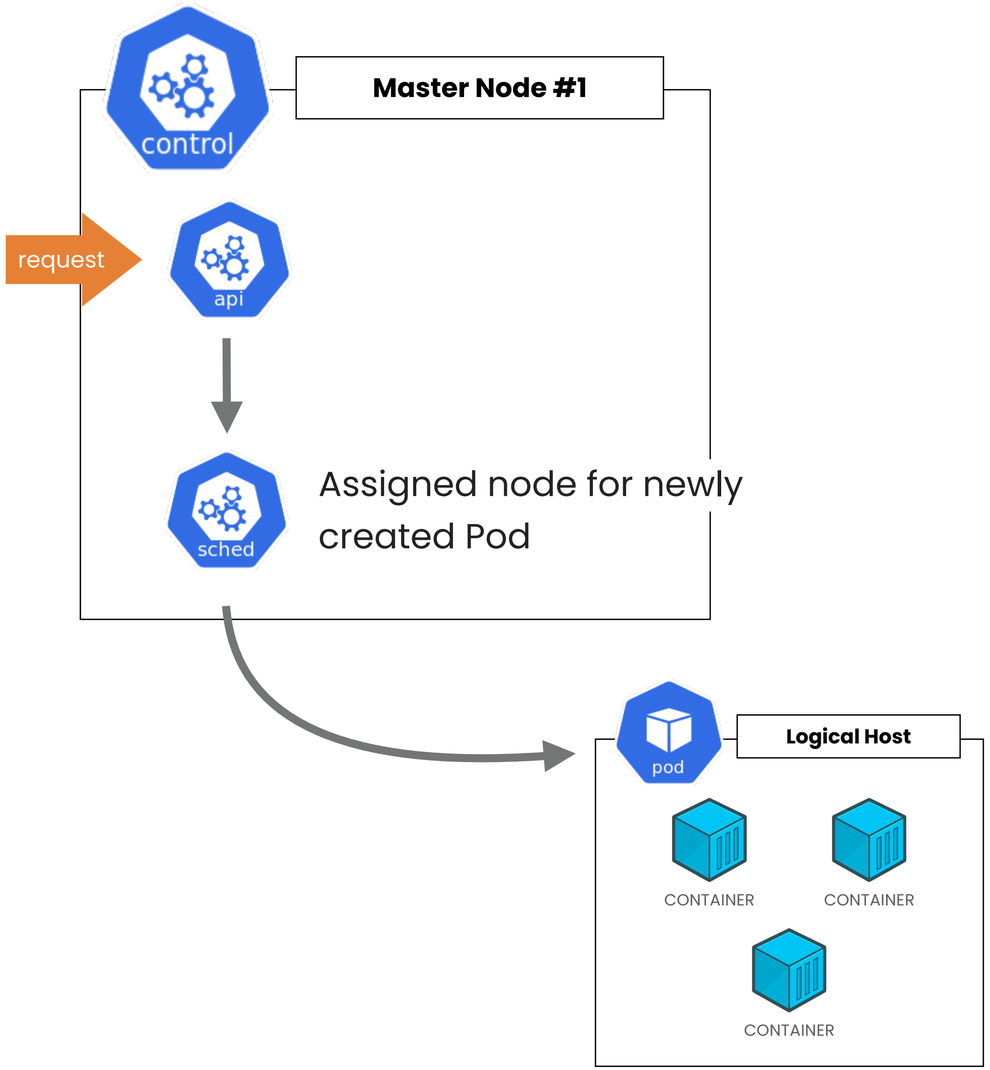
Control plane component that watches for newly created Pods with no assigned node, and selects a node for them to run on.

MASTER NODE - KUBE-SCHEDULER



Control plane component that watches for newly created Pods with no assigned node, and selects a node for them to run on.

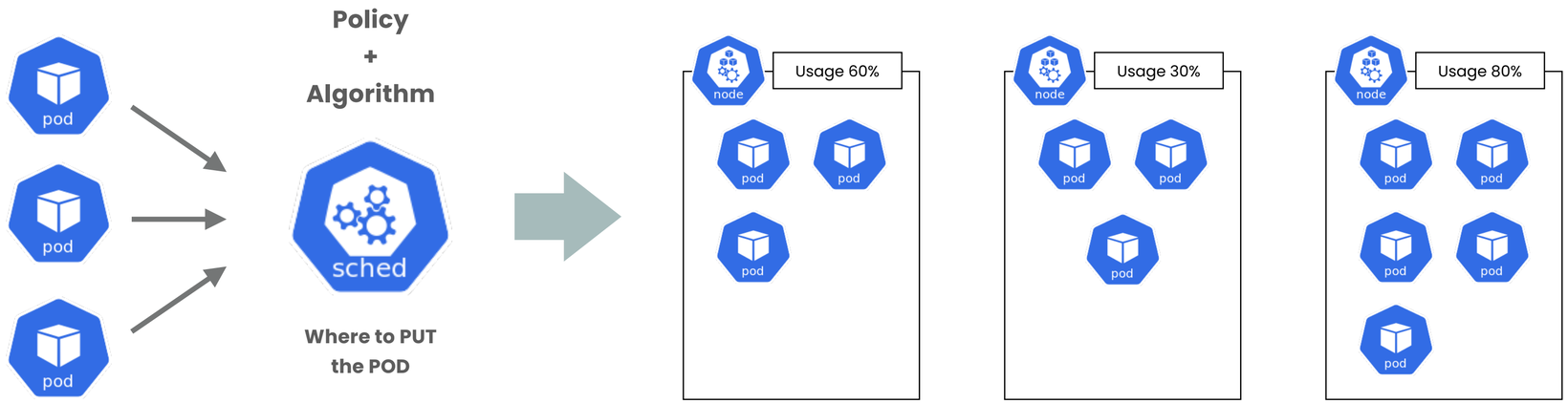
MASTER NODE - KUBE-SCHEDULER



Control plane component that watches for newly created Pods with no assigned node, and selects a node for them to run on.

MASTER NODE - KUBE-SCHEDULER

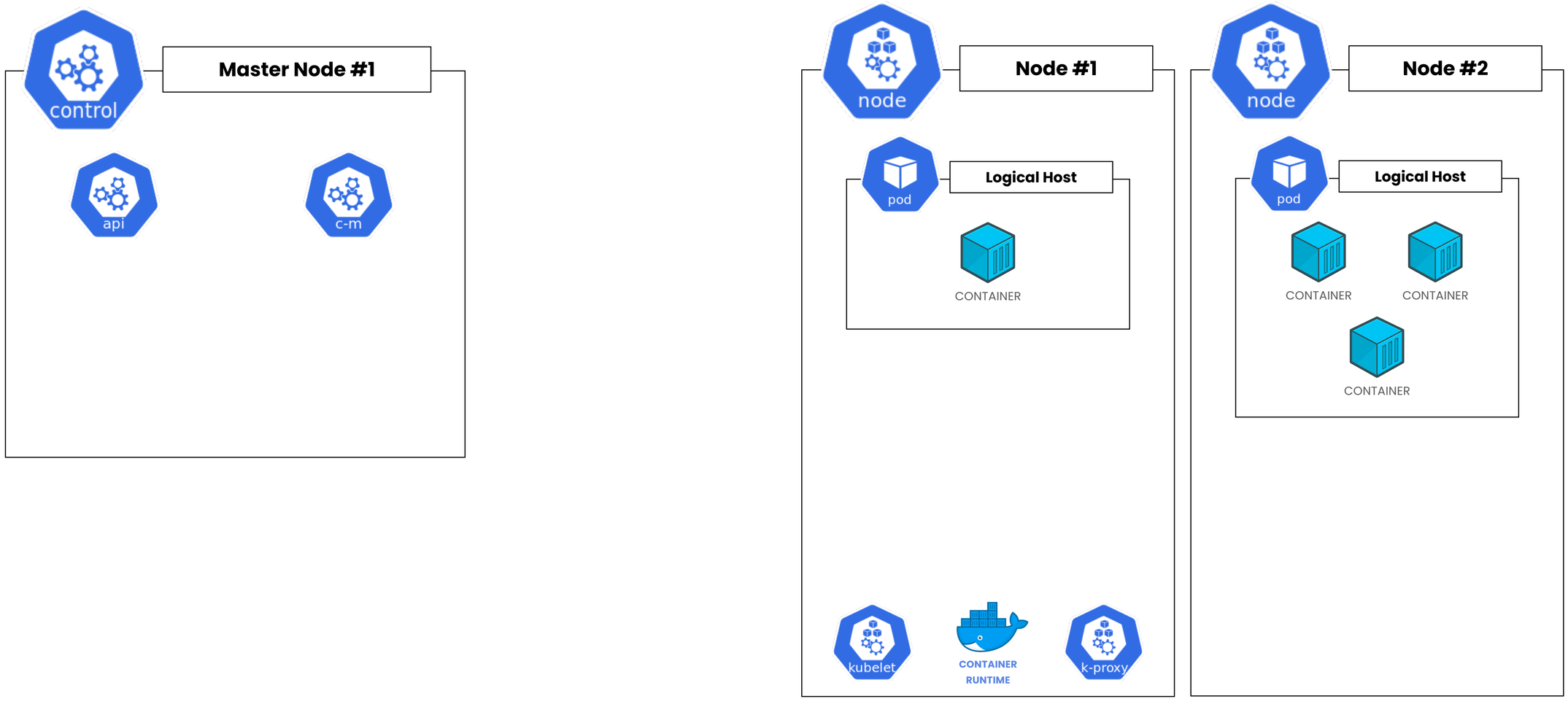
Factors taken into account for scheduling decisions include: individual and collective resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference, and deadlines.



MASTER NODE - KUBE-CONTROLLER-MANAGER

Control plane component that runs controller processes.

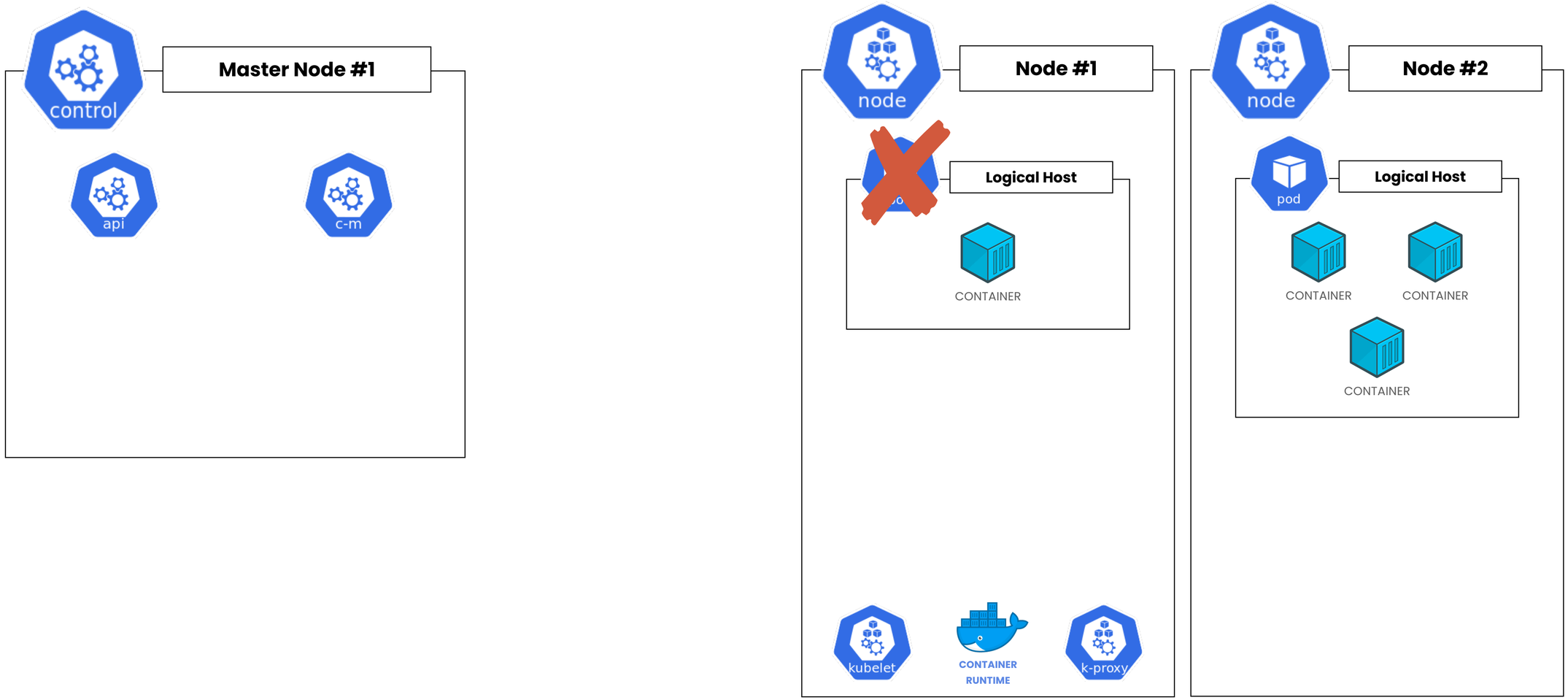
Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.



MASTER NODE - KUBE-CONTROLLER-MANAGER

Control plane component that runs controller processes.

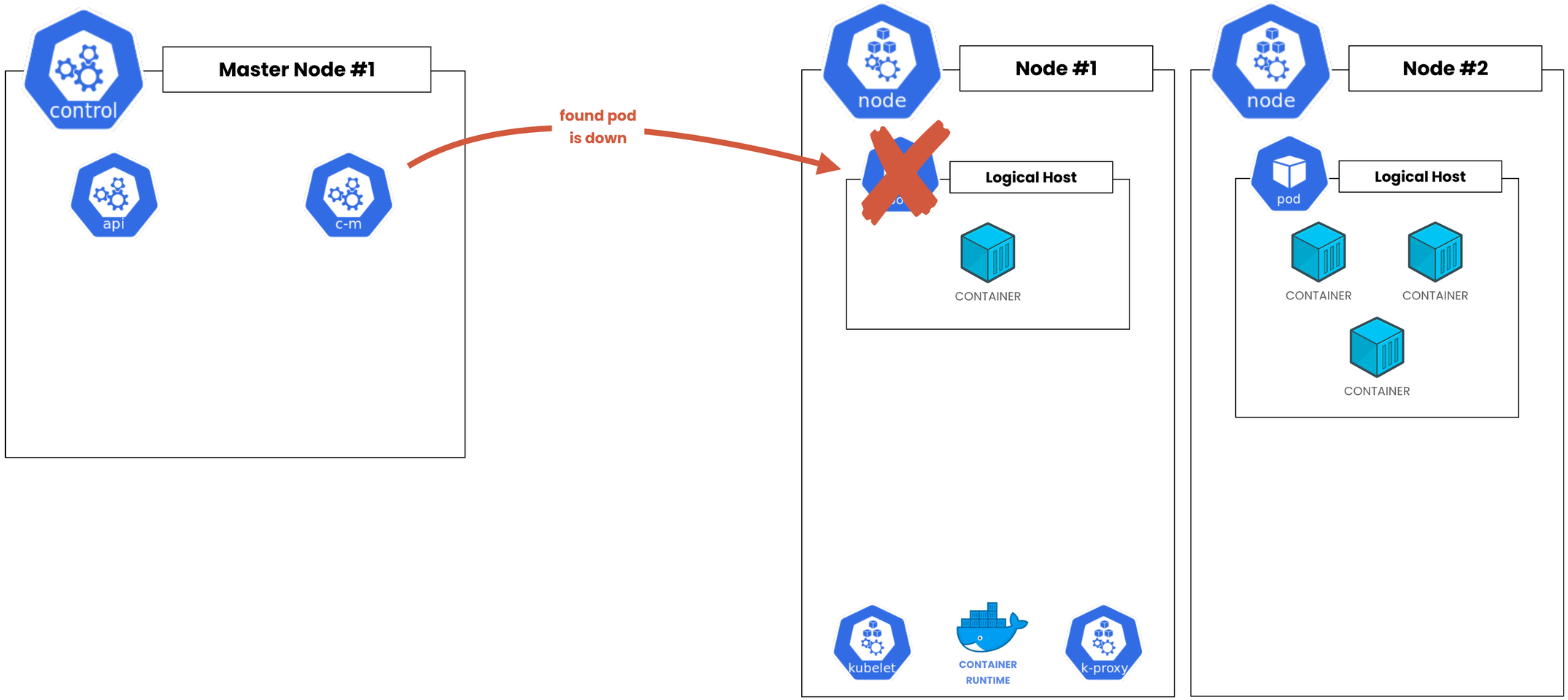
Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.



MASTER NODE - KUBE-CONTROLLER-MANAGER

Control plane component that runs controller processes.

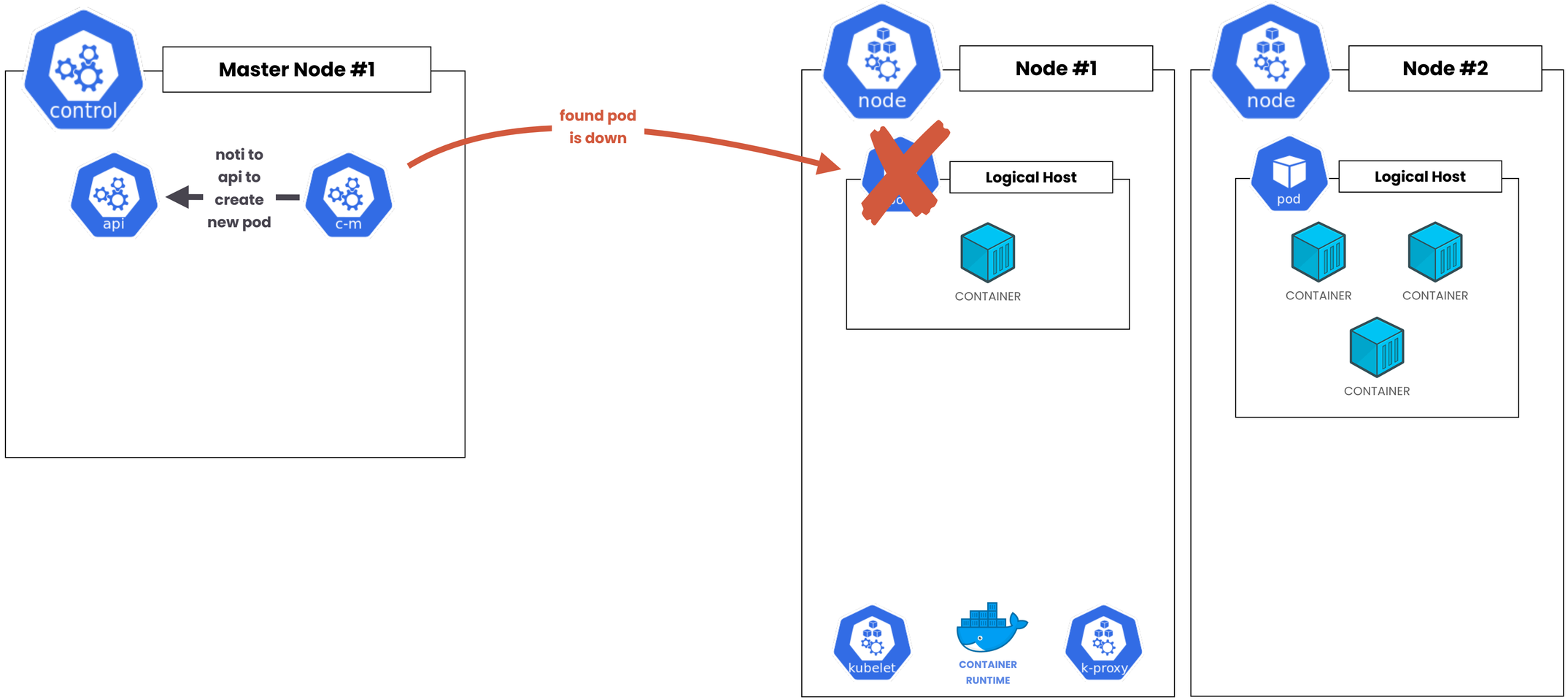
Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.



MASTER NODE - KUBE-CONTROLLER-MANAGER

Control plane component that runs controller processes.

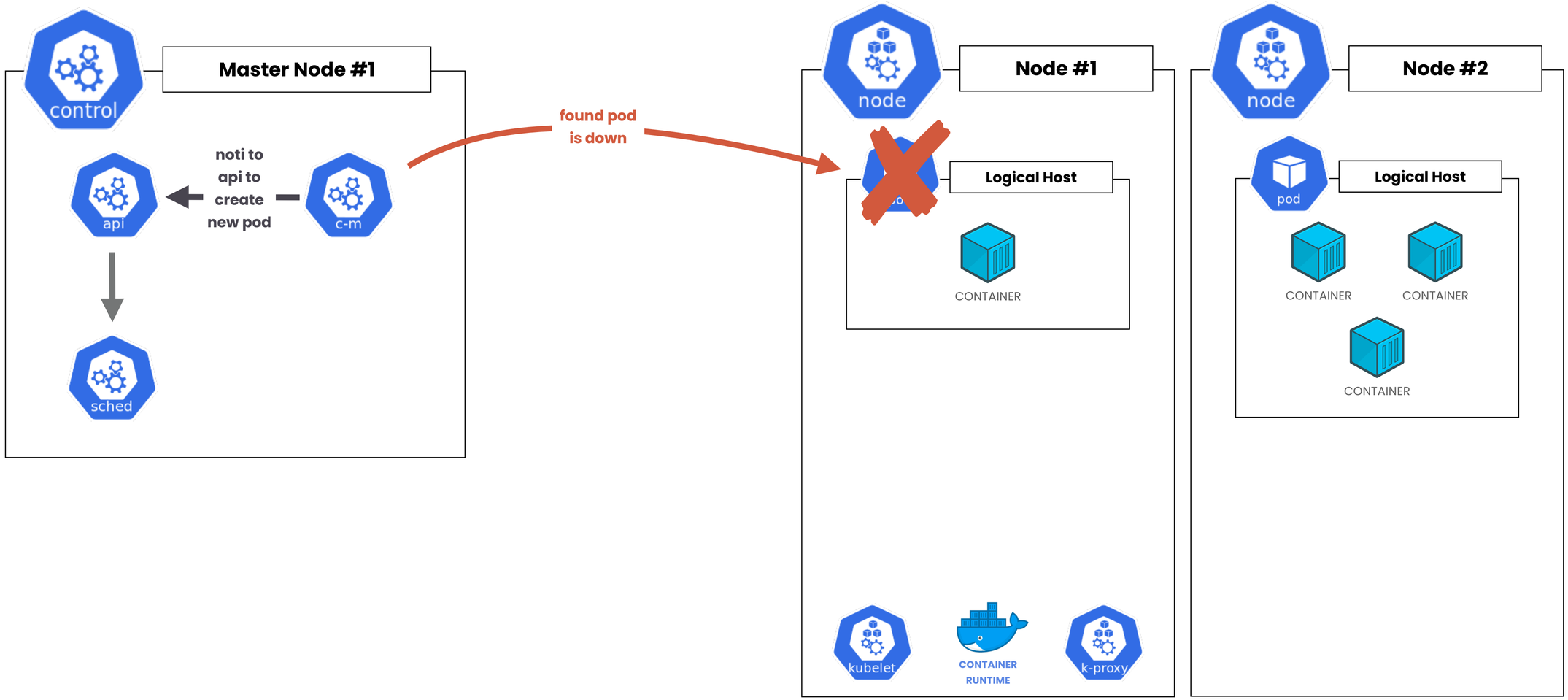
Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.



MASTER NODE - KUBE-CONTROLLER-MANAGER

Control plane component that runs controller processes.

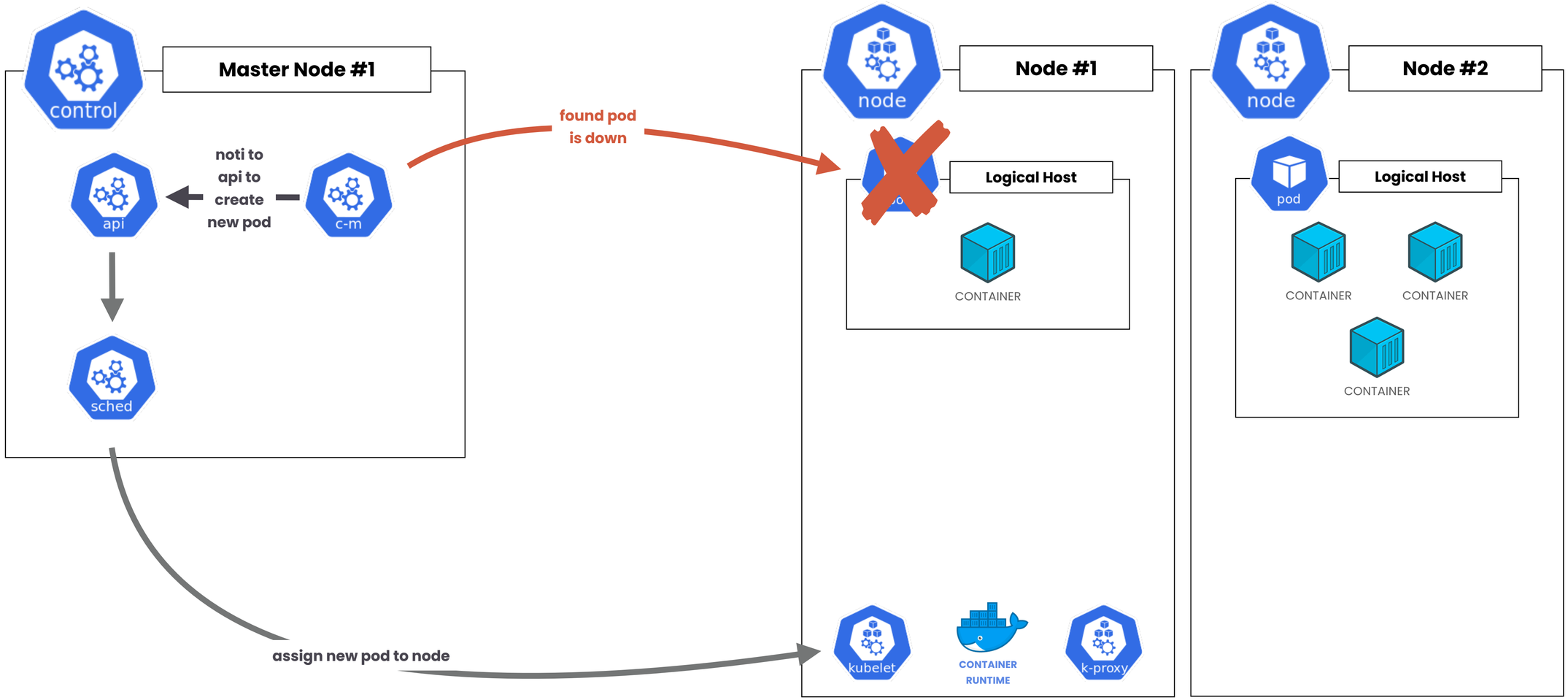
Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.



MASTER NODE - KUBE-CONTROLLER-MANAGER

Control plane component that runs controller processes.

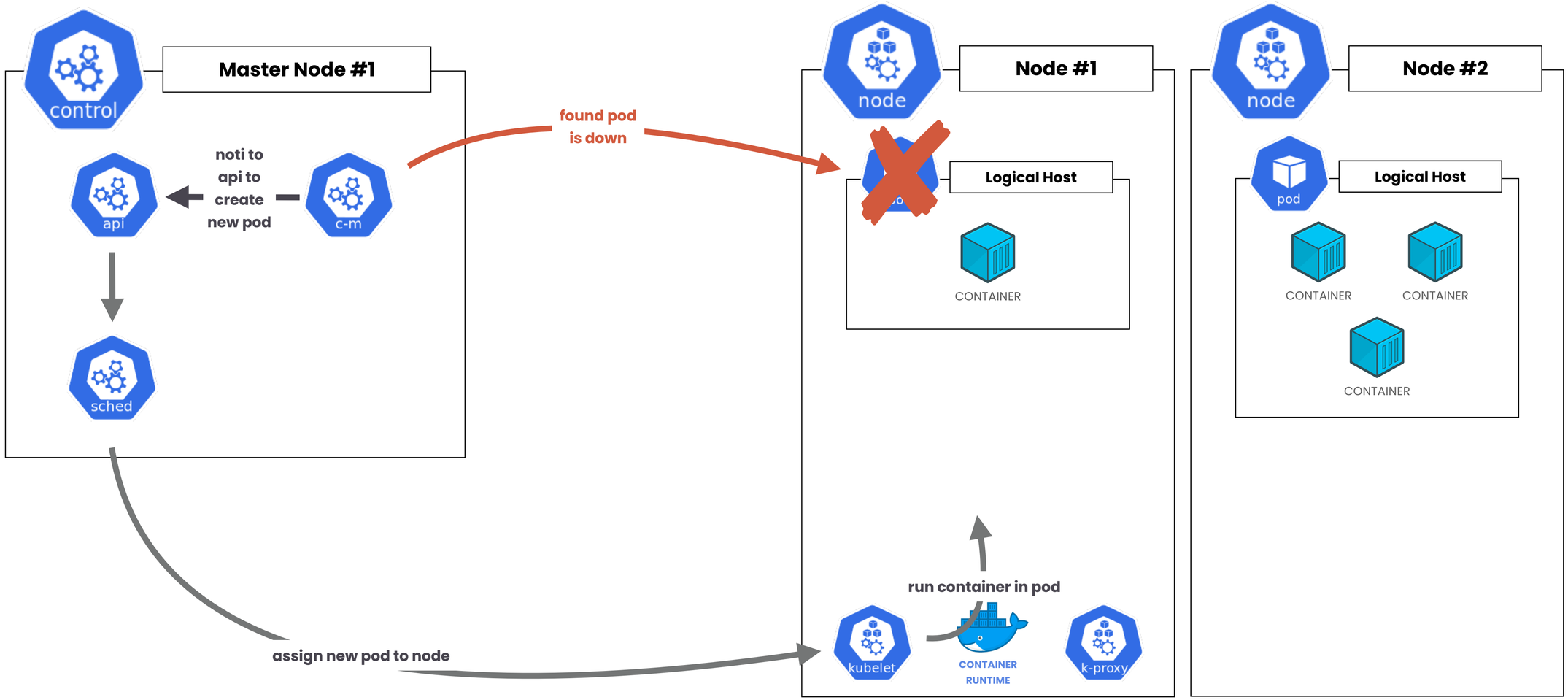
Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.



MASTER NODE - KUBE-CONTROLLER-MANAGER

Control plane component that runs controller processes.

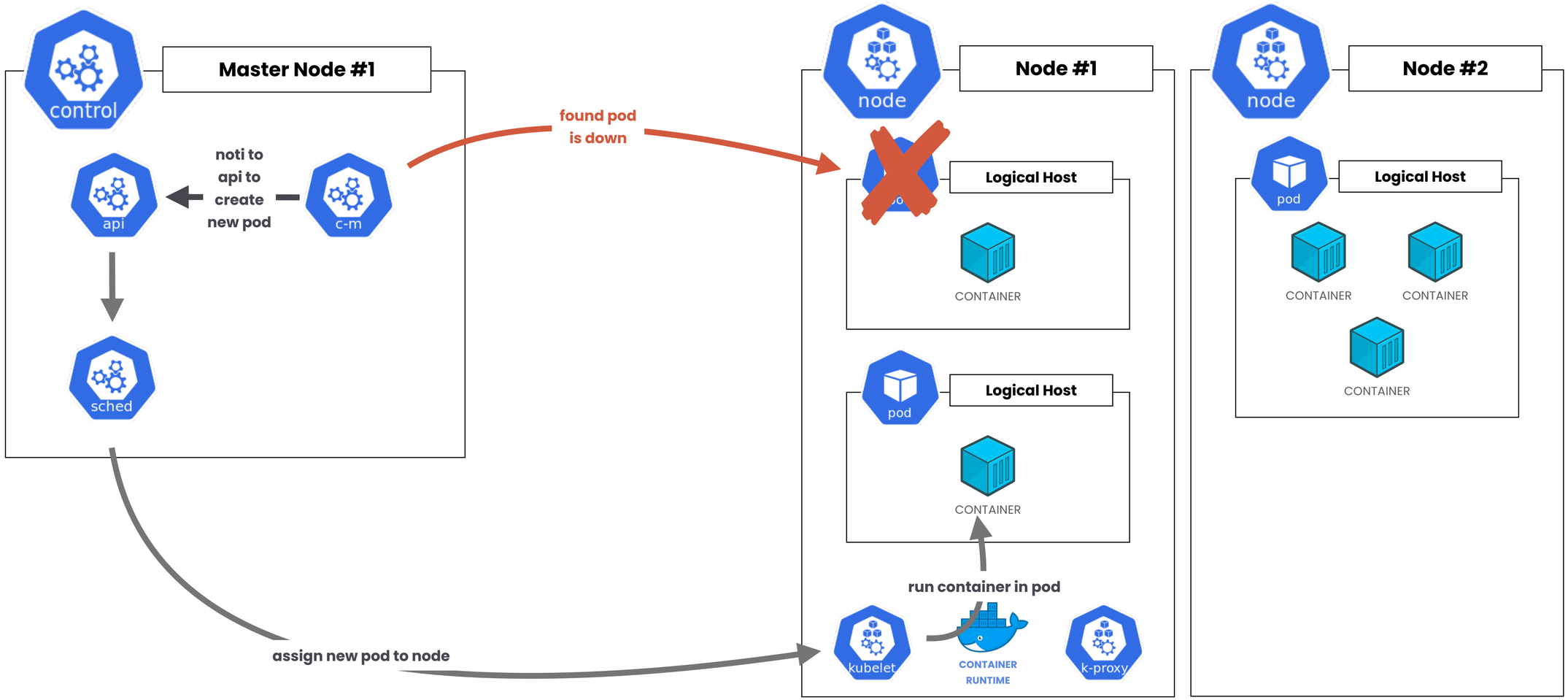
Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.



MASTER NODE - KUBE-CONTROLLER-MANAGER

Control plane component that runs controller processes.

Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.



MASTER NODE - KUBE-CONTROLLER-MANAGER

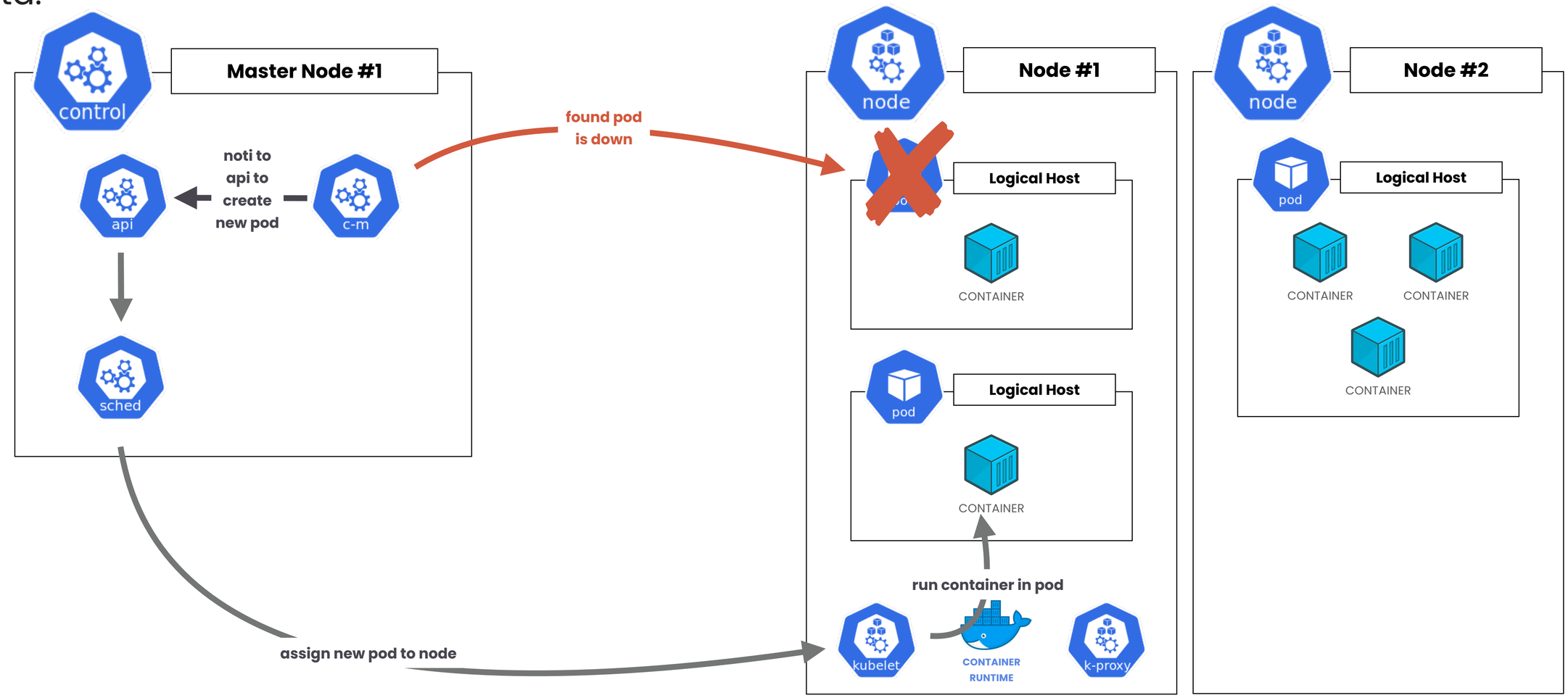
Some types of these controllers are:

- **Node controller**: Responsible for noticing and responding when nodes go down.
- **Job controller**: Watches for Job objects that represent one-off tasks, then creates Pods to run those tasks to completion.
- **Endpoints controller**: Populates the Endpoints object (that is, joins Services & Pods).
- **Service Account & Token controllers**: Create default accounts and API access tokens for new namespaces.

MASTER NODE - ETCD

Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

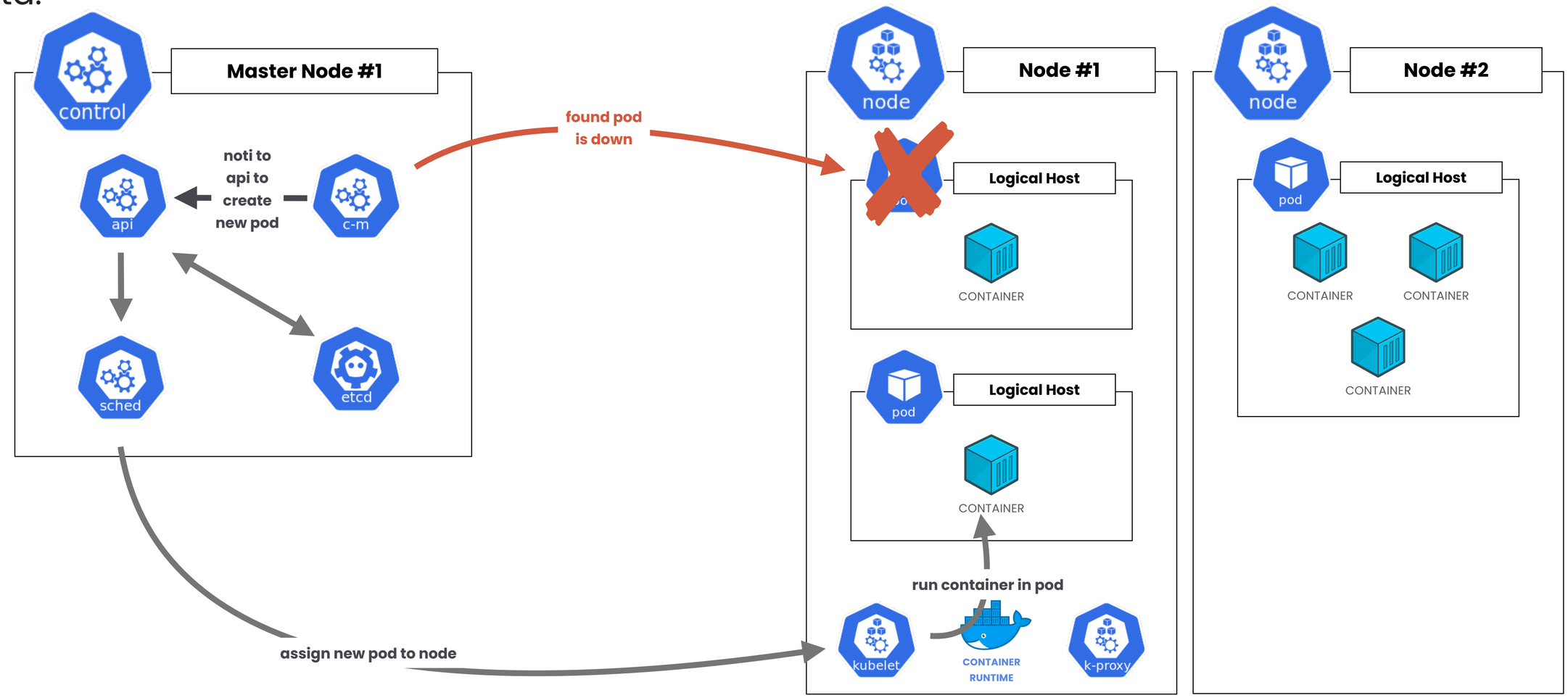
If your Kubernetes cluster uses etcd as its backing store, make sure you have a back up plan for those data.



MASTER NODE - ETCD

Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

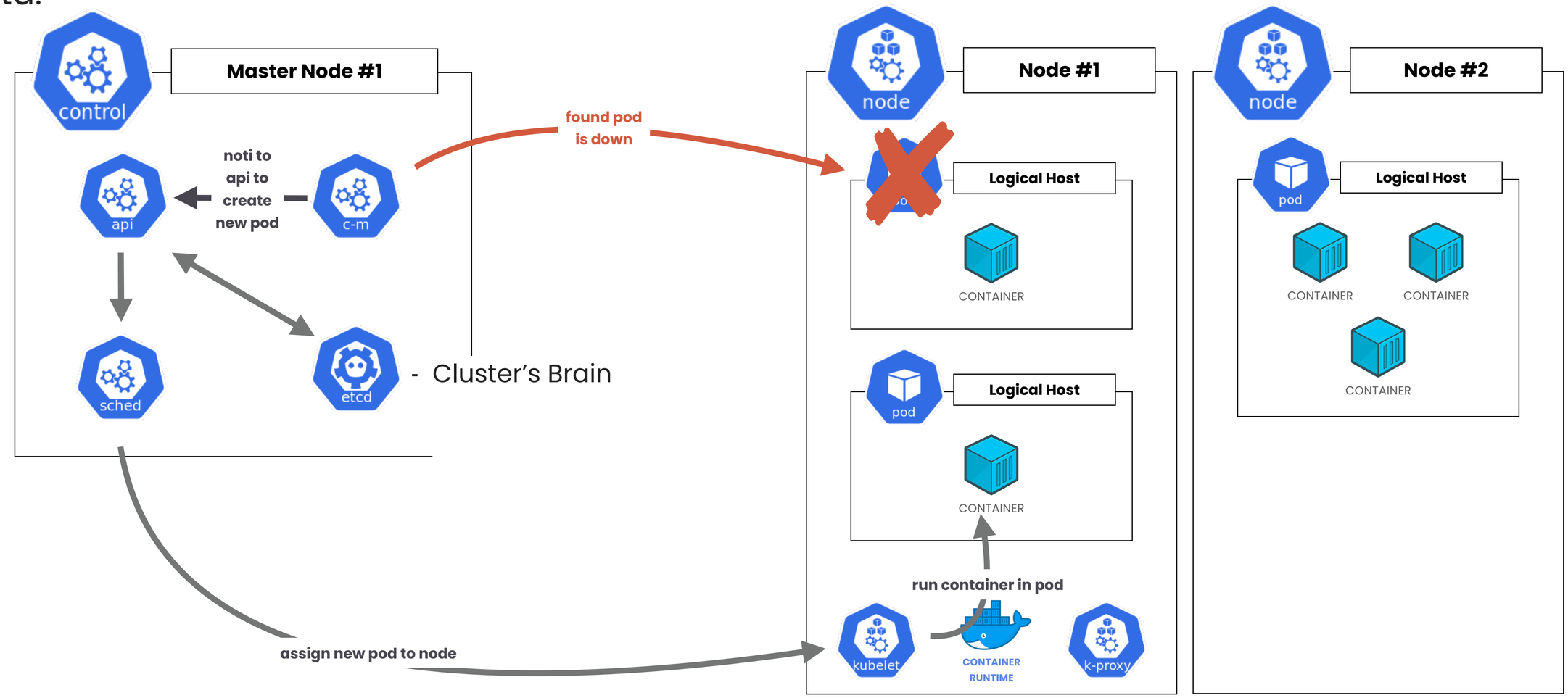
If your Kubernetes cluster uses etcd as its backing store, make sure you have a back up plan for those data.



MASTER NODE - ETCD

Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

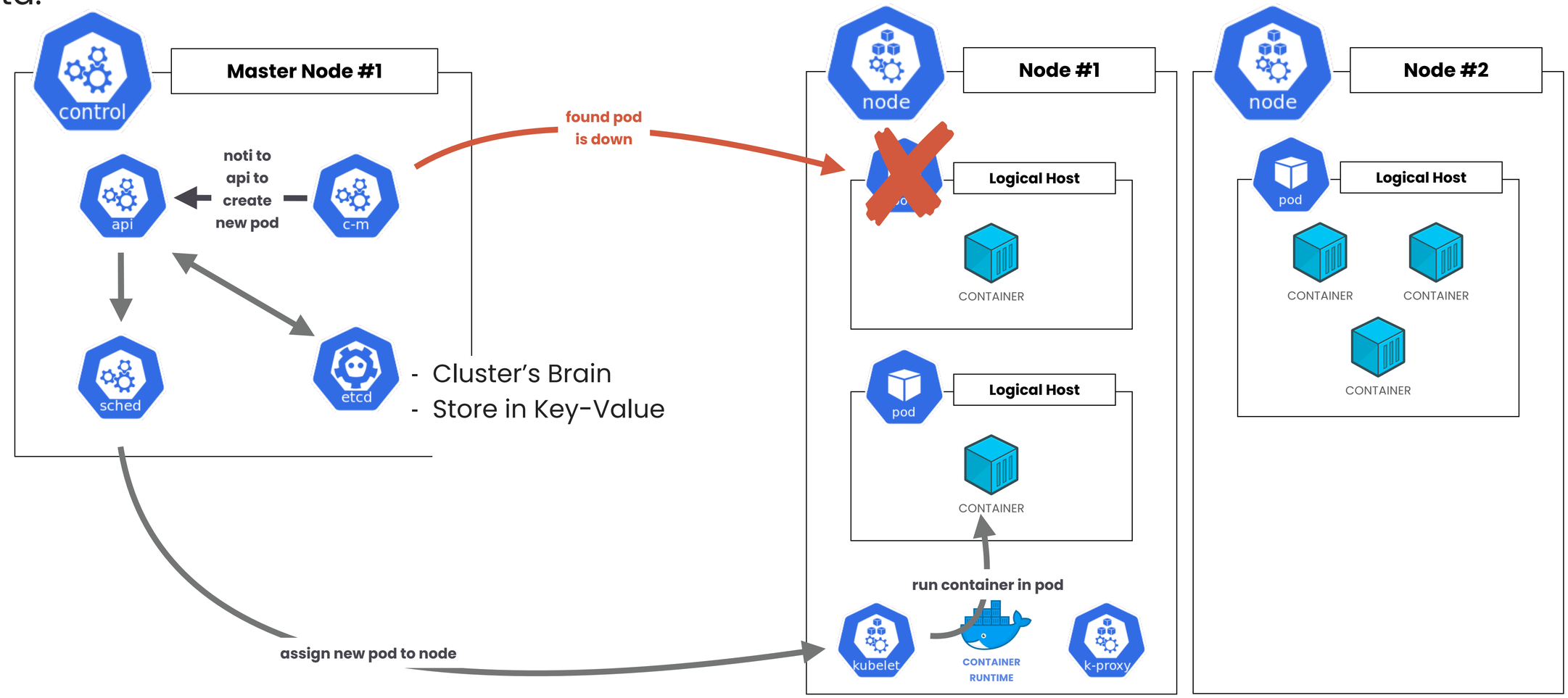
If your Kubernetes cluster uses etcd as its backing store, make sure you have a back up plan for those data.



MASTER NODE - ETCD

Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

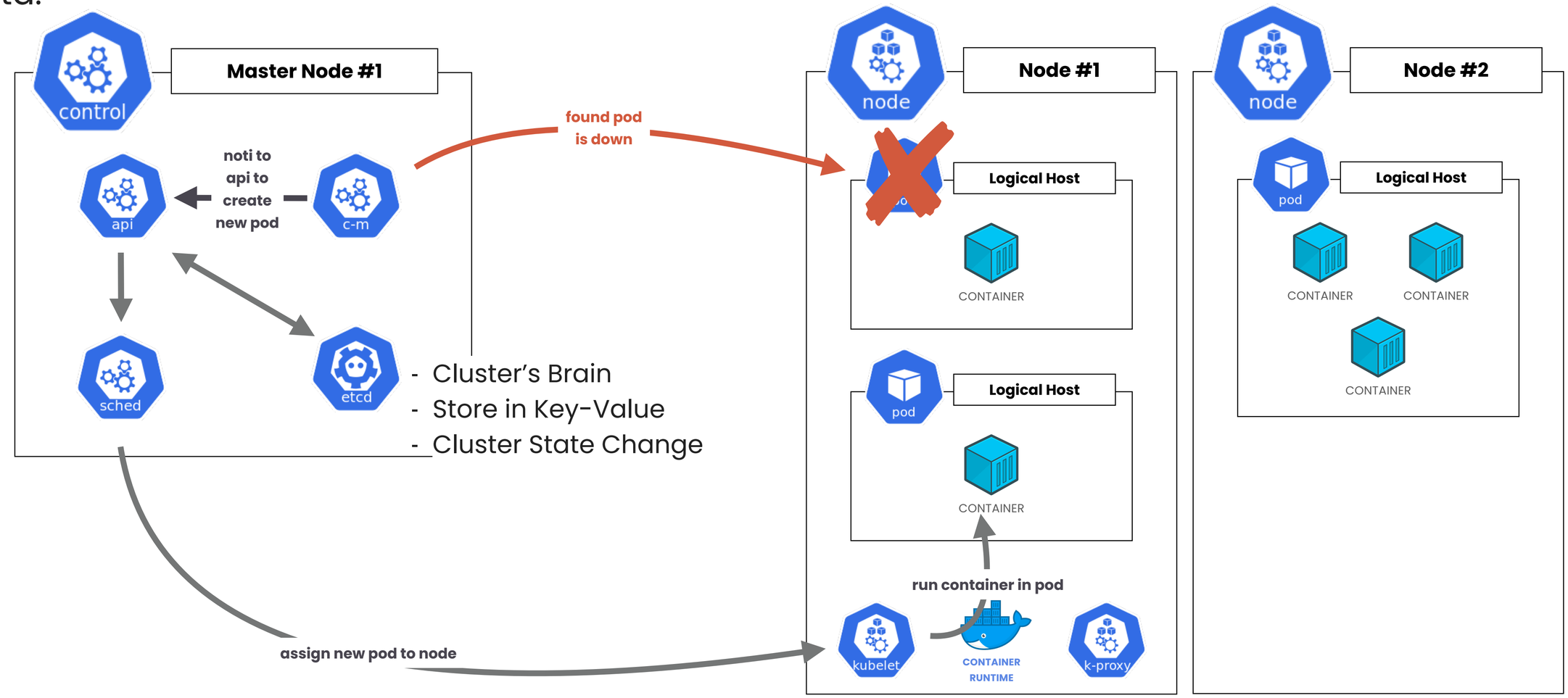
If your Kubernetes cluster uses etcd as its backing store, make sure you have a back up plan for those data.



MASTER NODE - ETCD

Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

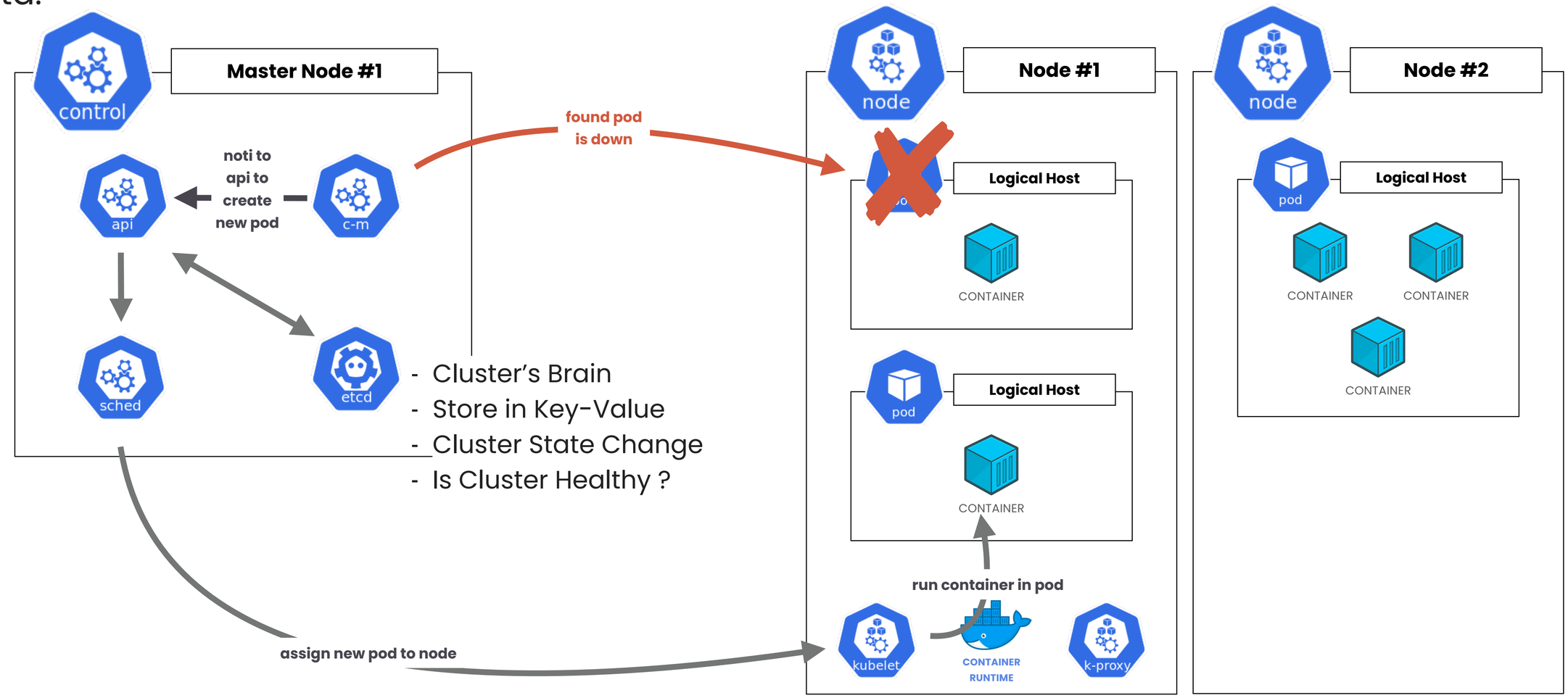
If your Kubernetes cluster uses etcd as its backing store, make sure you have a back up plan for those data.



MASTER NODE - ETCD

Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

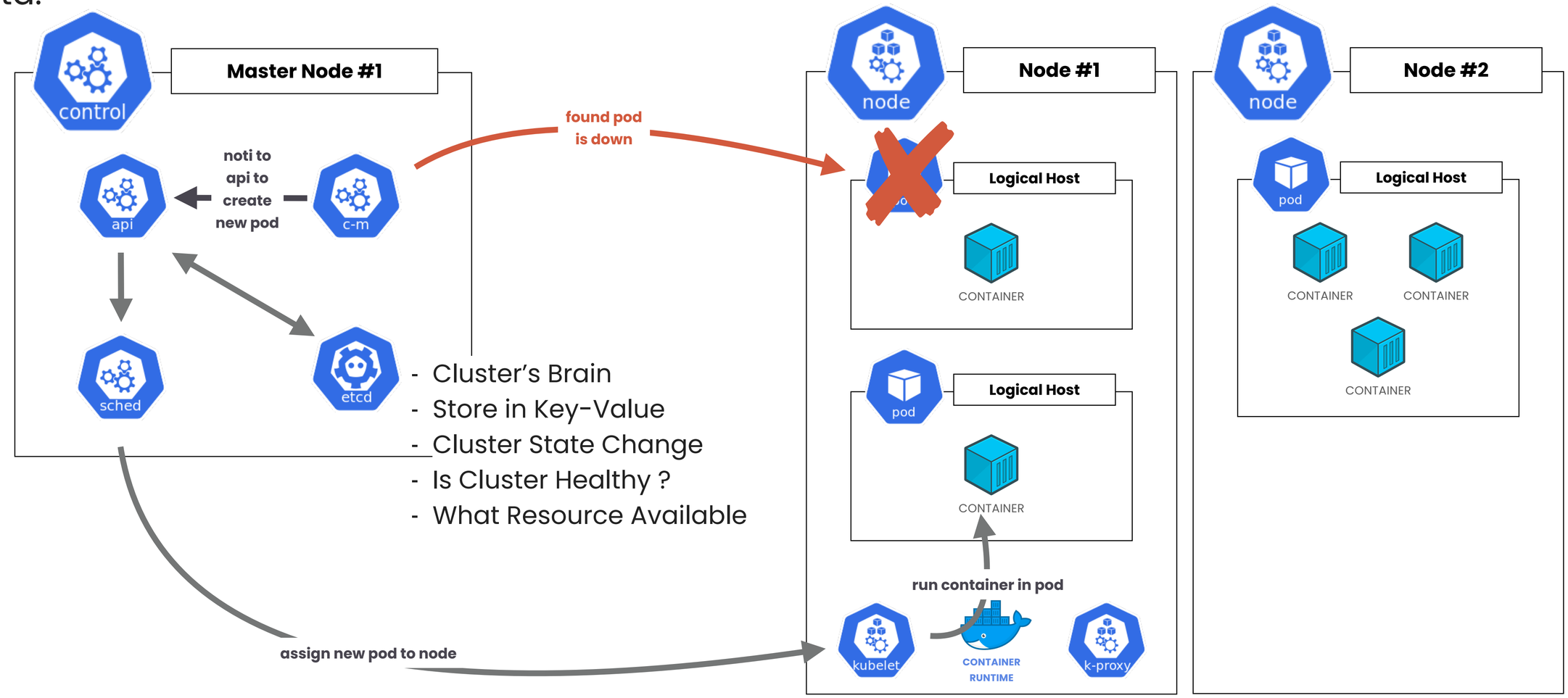
If your Kubernetes cluster uses etcd as its backing store, make sure you have a back up plan for those data.



MASTER NODE - ETCD

Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

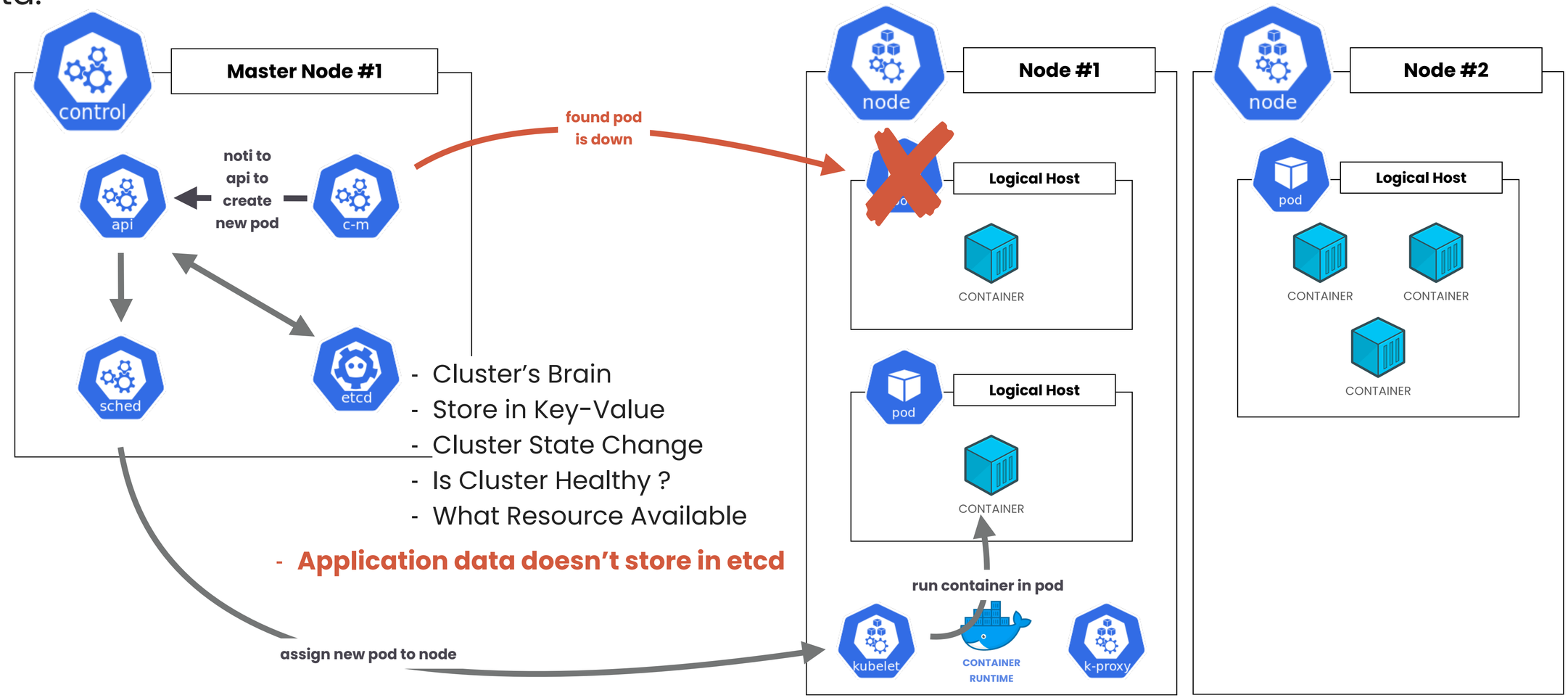
If your Kubernetes cluster uses etcd as its backing store, make sure you have a back up plan for those data.



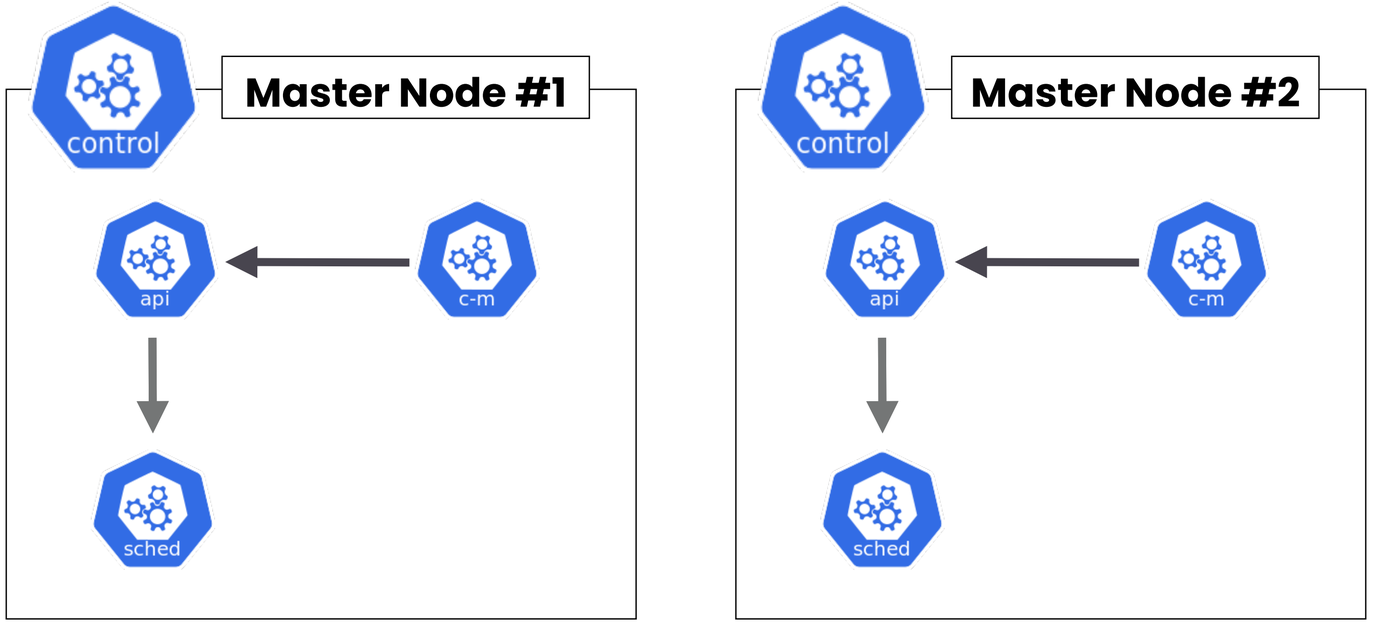
MASTER NODE - ETCD

Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

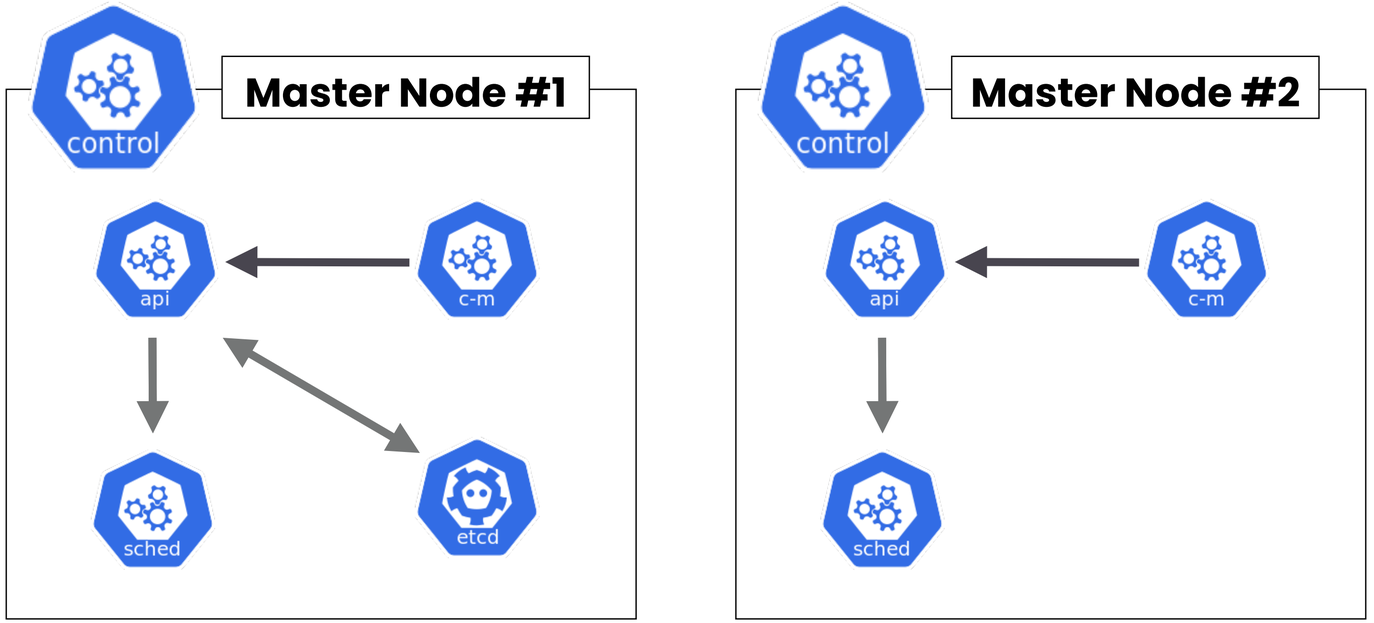
If your Kubernetes cluster uses etcd as its backing store, make sure you have a back up plan for those data.



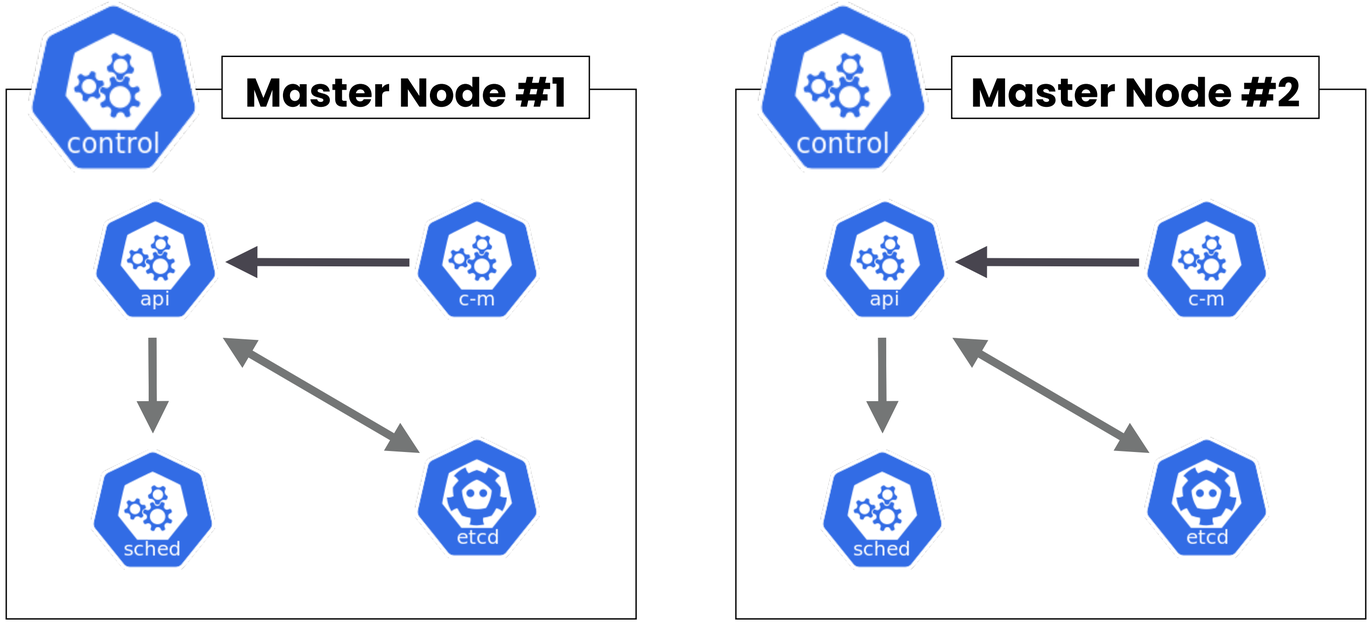
MASTER NODE - REDUNDANCY



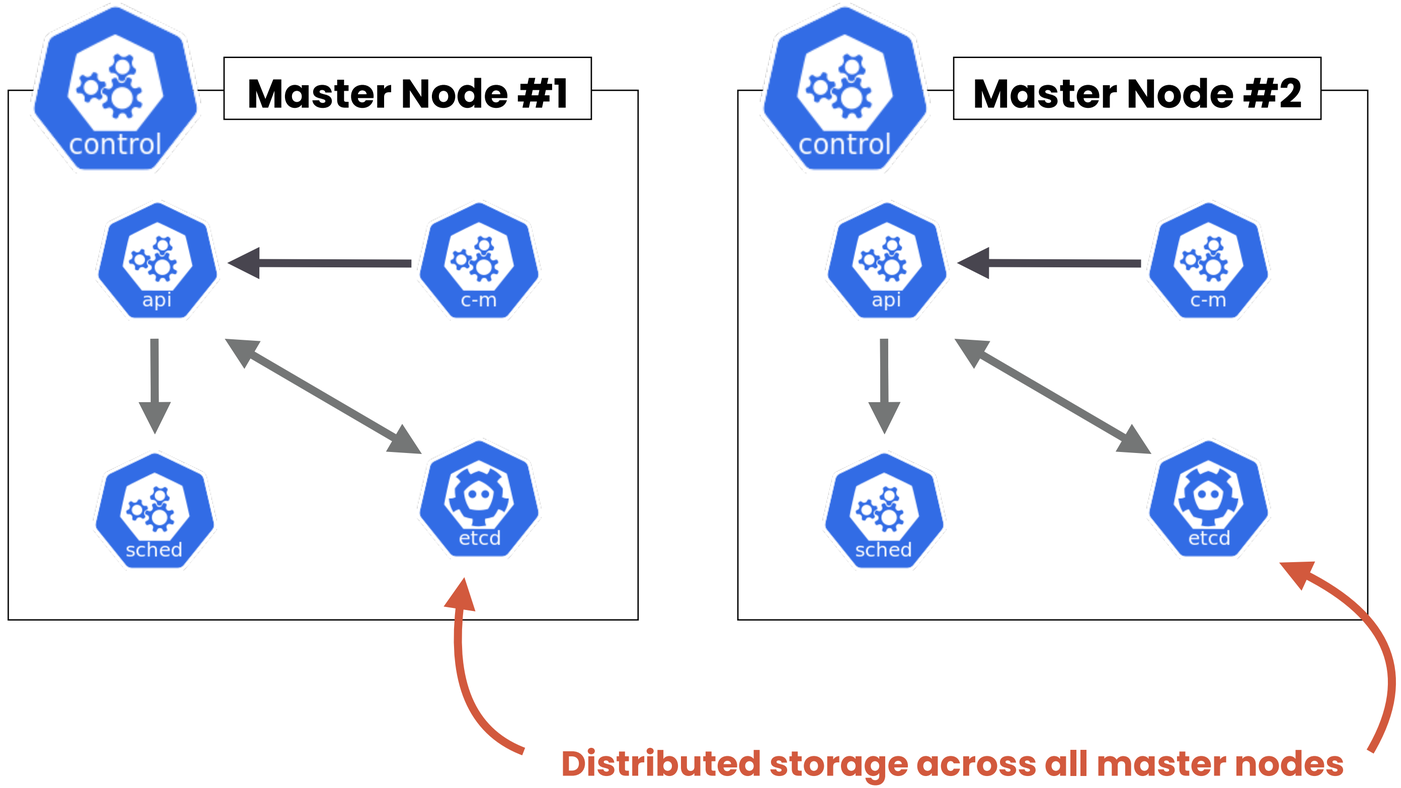
MASTER NODE - REDUNDANCY



MASTER NODE - REDUNDANCY

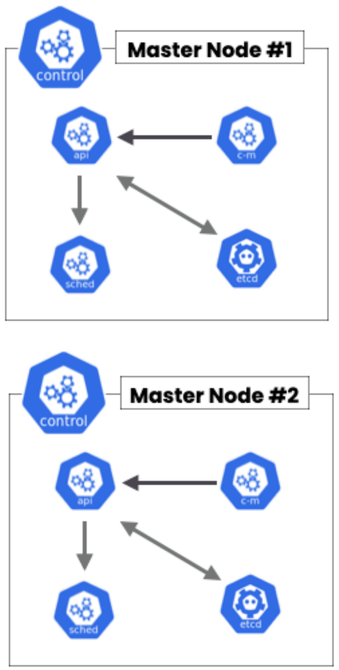


MASTER NODE - REDUNDANCY



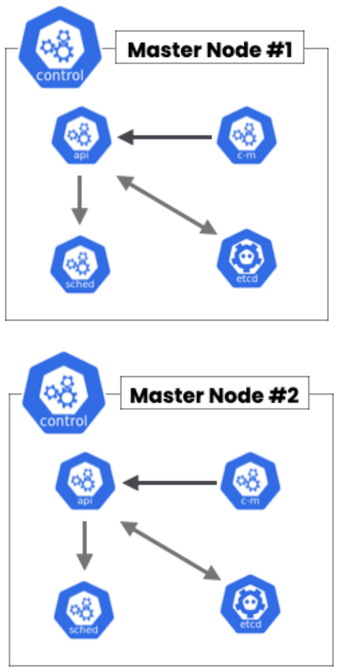
EXAMPLE – CLUSTER SETUP

EXAMPLE – CLUSTER SETUP



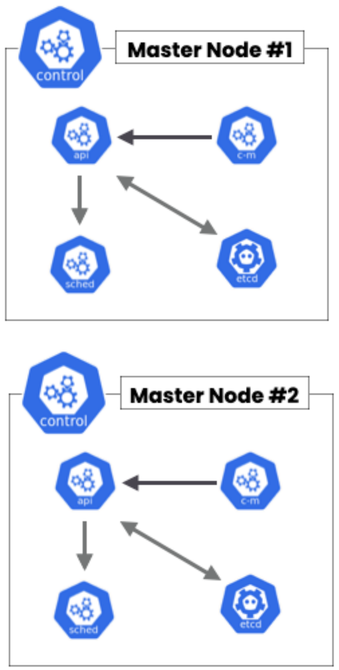
EXAMPLE - CLUSTER SETUP

LESS RESOURCES



EXAMPLE - CLUSTER SETUP

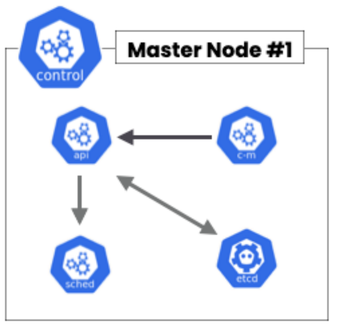
LESS RESOURCES

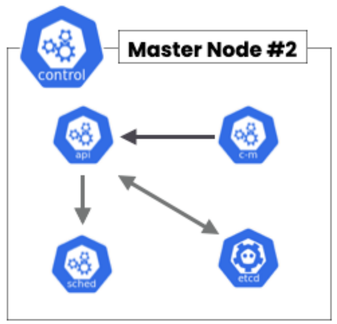


MORE STABLE

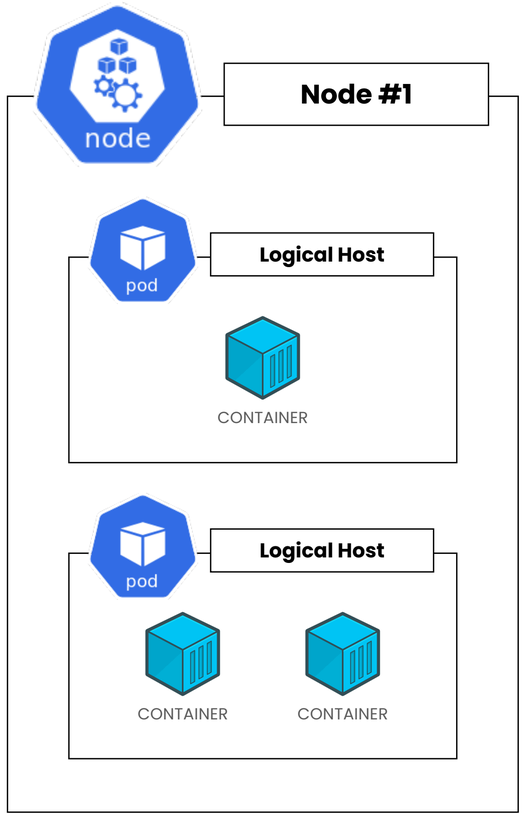
EXAMPLE - CLUSTER SETUP

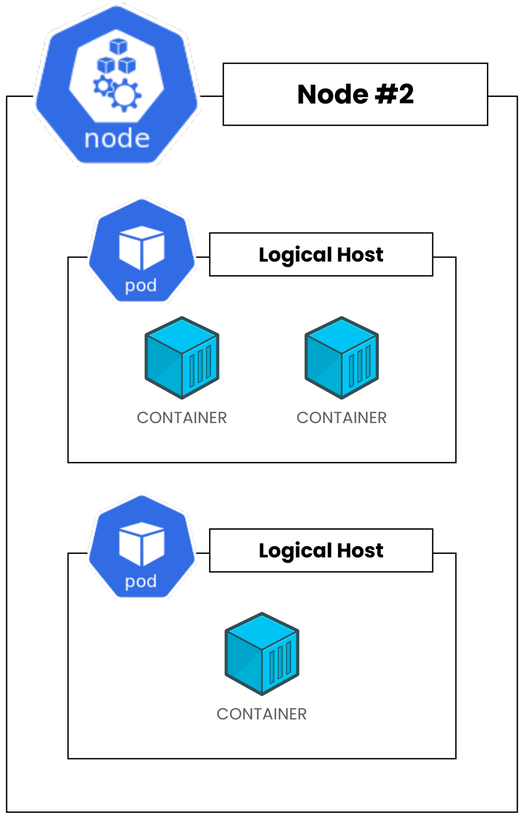
LESS RESOURCES

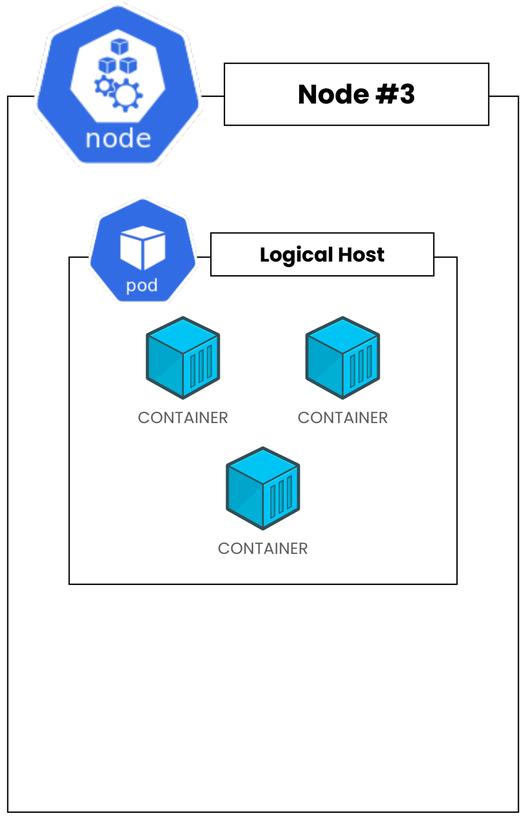




MORE STABLE

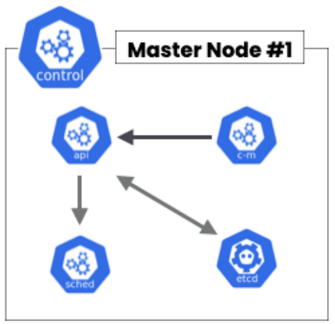


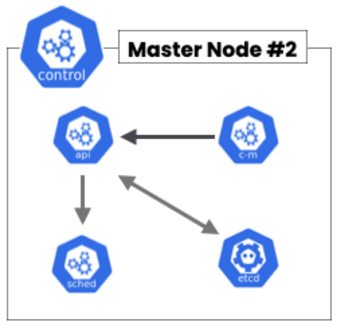




EXAMPLE - CLUSTER SETUP

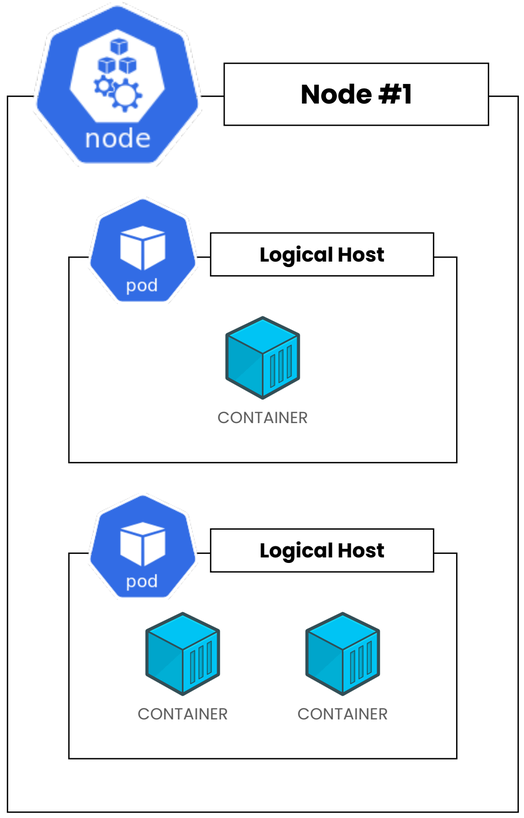
LESS RESOURCES

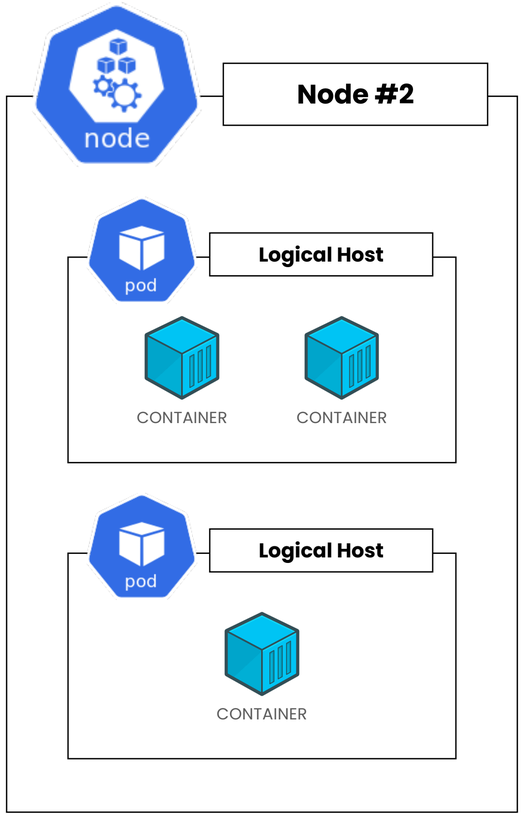


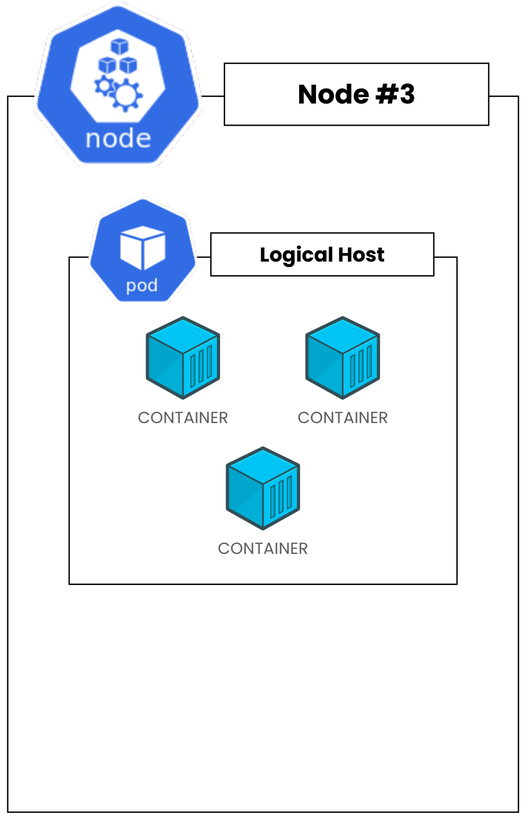


MORE STABLE

MORE RESOURCES

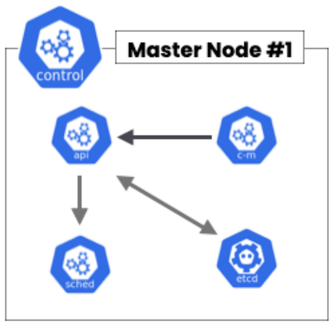


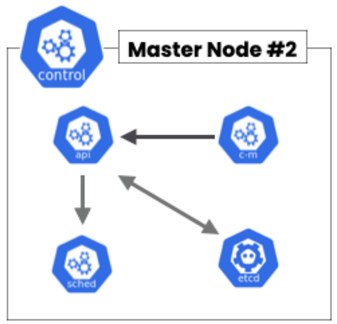




EXAMPLE - CLUSTER SETUP

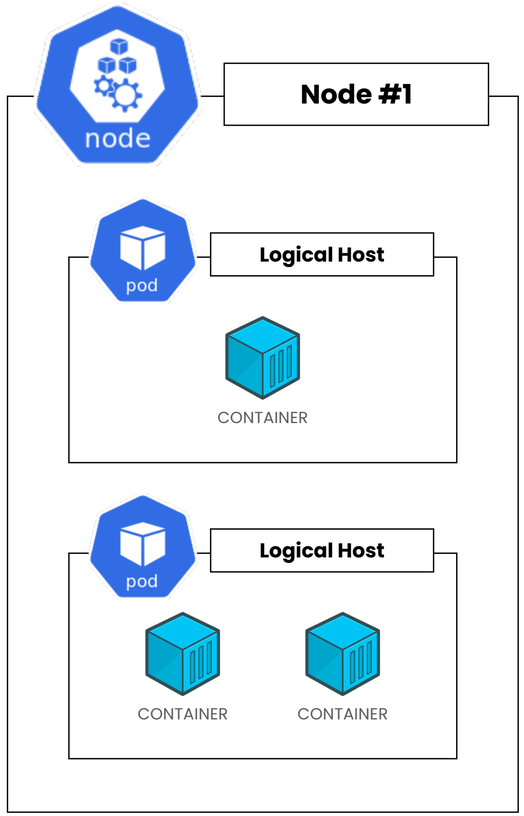
LESS RESOURCES

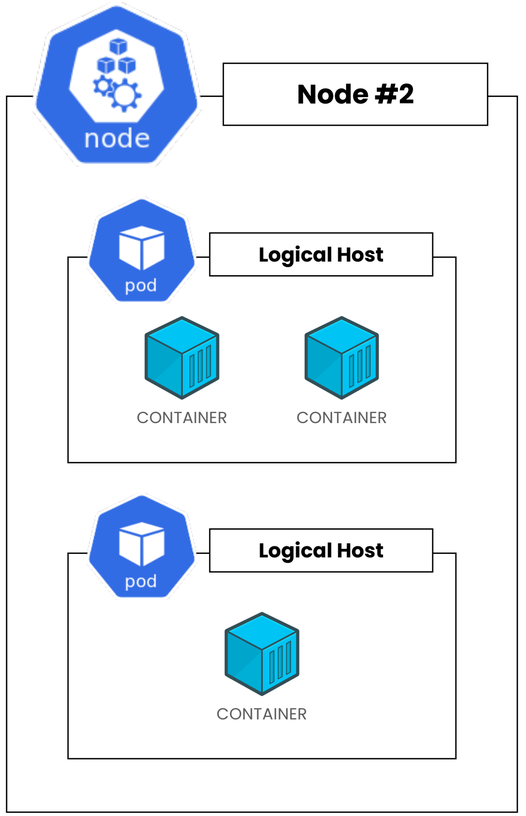


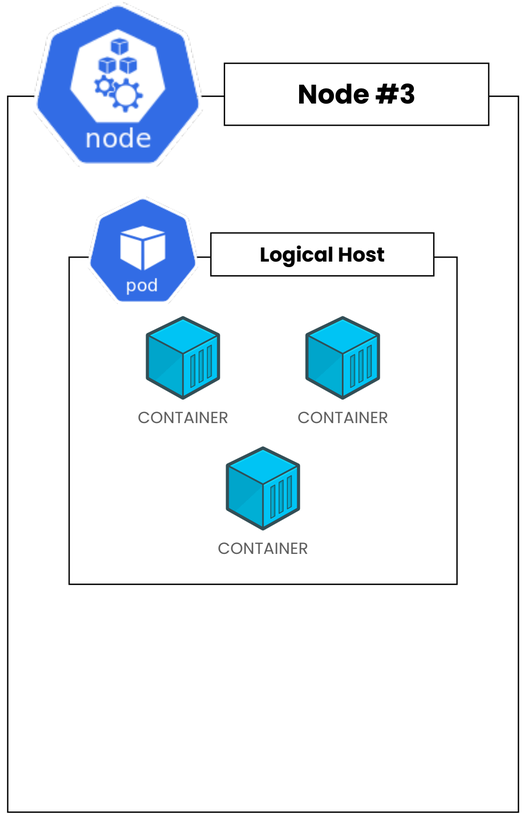


MORE STABLE

MORE RESOURCES







CAN REPLACE

MASTER NODE - CLOUD-CONTROLLER-MANAGER

A Kubernetes control plane component that embeds cloud-specific control logic. The cloud controller manager lets you link your cluster into your cloud provider's API, and separates out the components that interact with that cloud platform from components that only interact with your cluster.

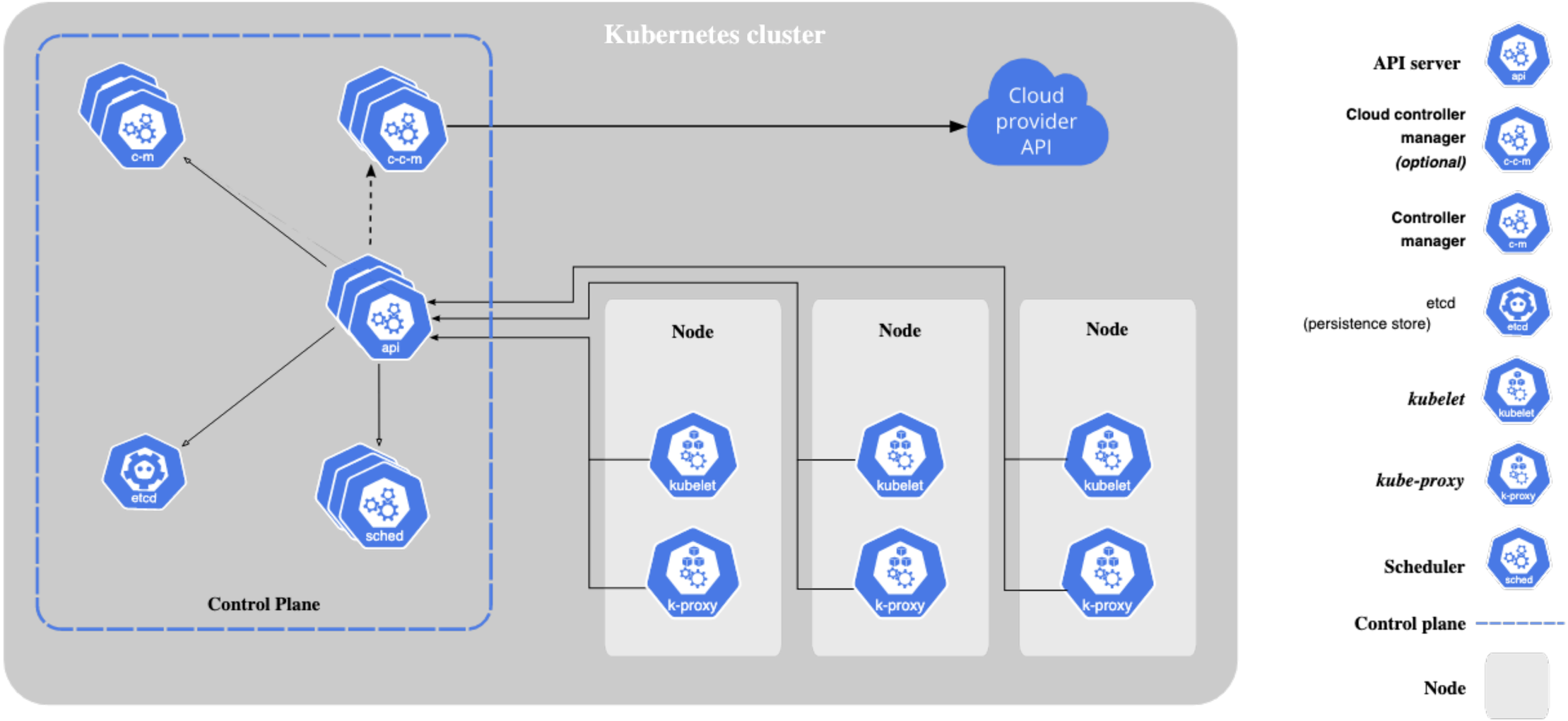
The cloud-controller-manager only runs controllers that are specific to your cloud provider. If you are running Kubernetes on your own premises, or in a learning environment inside your own PC, the cluster does not have a cloud controller manager.

As with the kube-controller-manager, the cloud-controller-manager combines several logically independent control loops into a single binary that you run as a single process. You can scale horizontally (run more than one copy) to improve performance or to help tolerate failures.

The following controllers can have cloud provider dependencies:

- Node controller: For checking the cloud provider to determine if a node has been deleted in the cloud after it stops responding
- Route controller: For setting up routes in the underlying cloud infrastructure
- Service controller: For creating, updating and deleting cloud provider load balancers

KUBERNETES ARCHITECTURE



THANK YOU