Kubernetes

Marcelo Schirbel Gomes

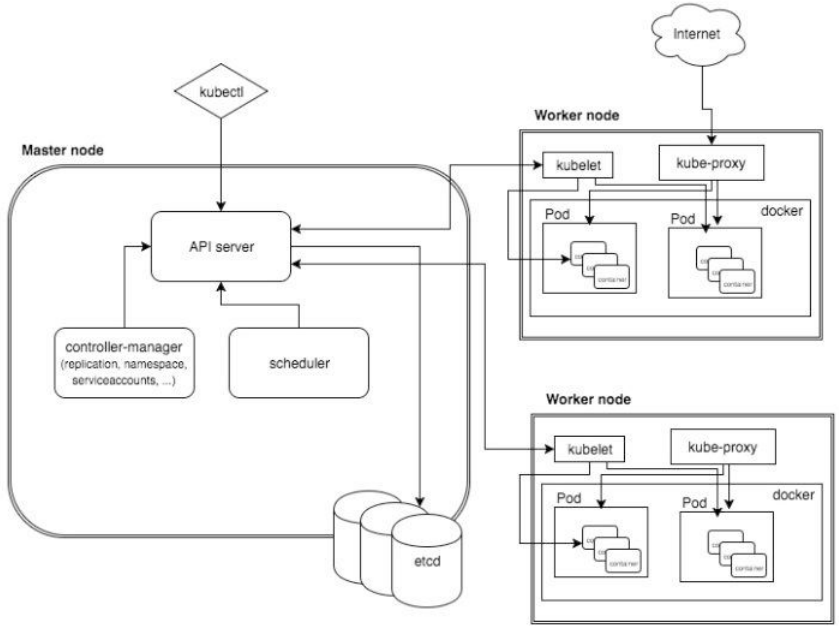
Today

1. Architecture
2. Components
3. Objects
4. Interactions
5. Demo
6. Exercices

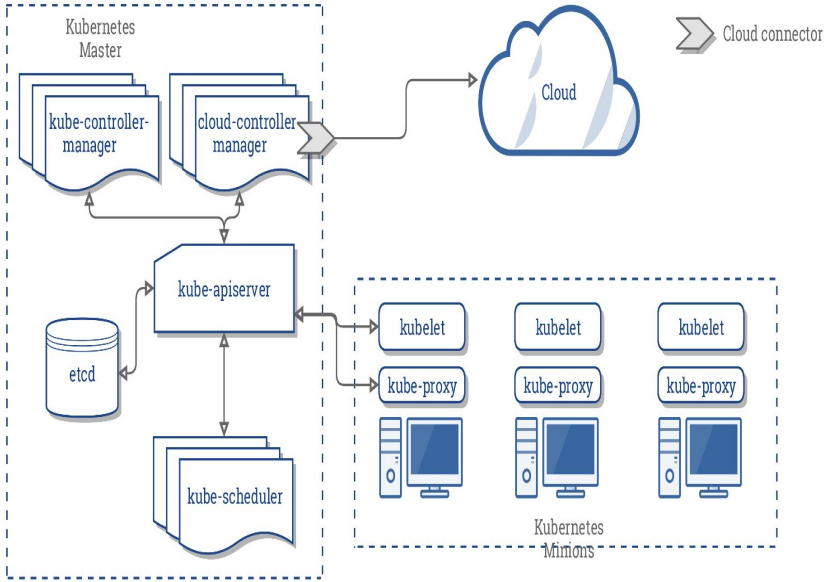
Architecture

The Kubernetes architecture is a distributed system that is designed to be highly available and scalable. It consists of a collection of Kubernetes nodes, each of which is responsible for a portion of the cluster. We have master nodes, which are responsible for managing the cluster, and worker nodes, which are responsible for running the workloads.

Since we are running on AWS EKS we don't have access to the Kubernetes master nodes.

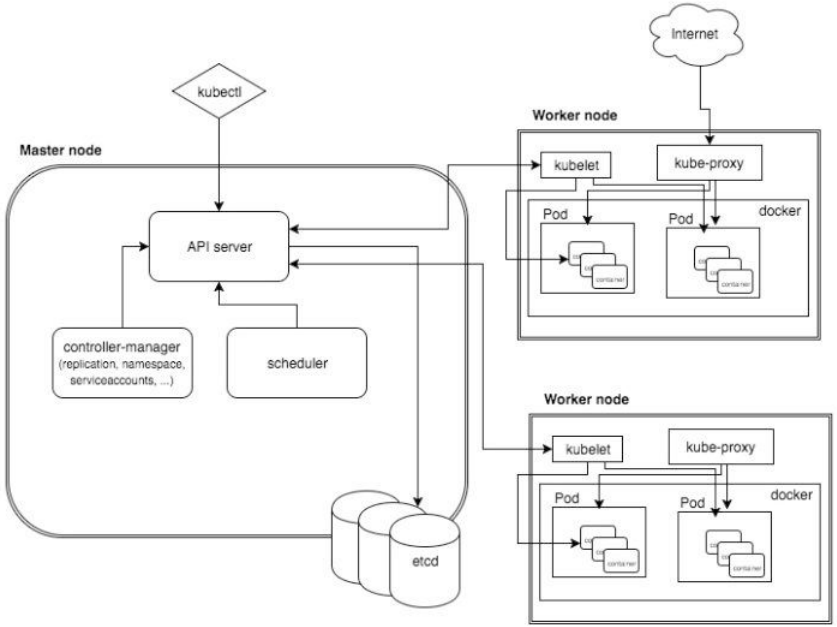


Components


Kubernetes
Master
kube-controller-
manager
cloud-controller
manager
Cloud
Cloud connector
kube-apiserver
etcd
kube-scheduler
kubelet
kubelet
kubelet
kube-proxy
kube-proxy
kube-proxy
Kubernetes
Minions

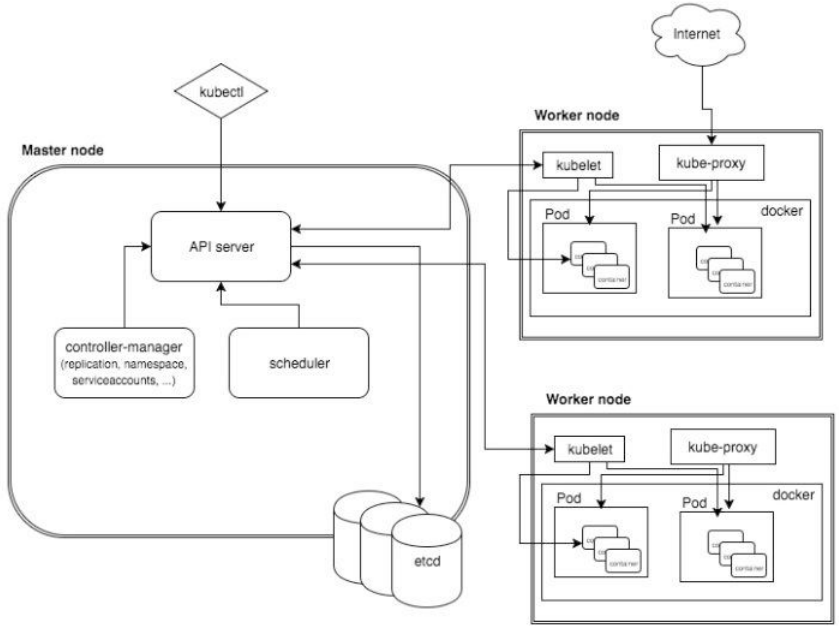
Kubelet

The kubelet is the primary component of Kubernetes. It is responsible for starting and maintaining the containers that make up your cluster. It can register with the Kubernetes API server and receive updates about the status of your cluster.



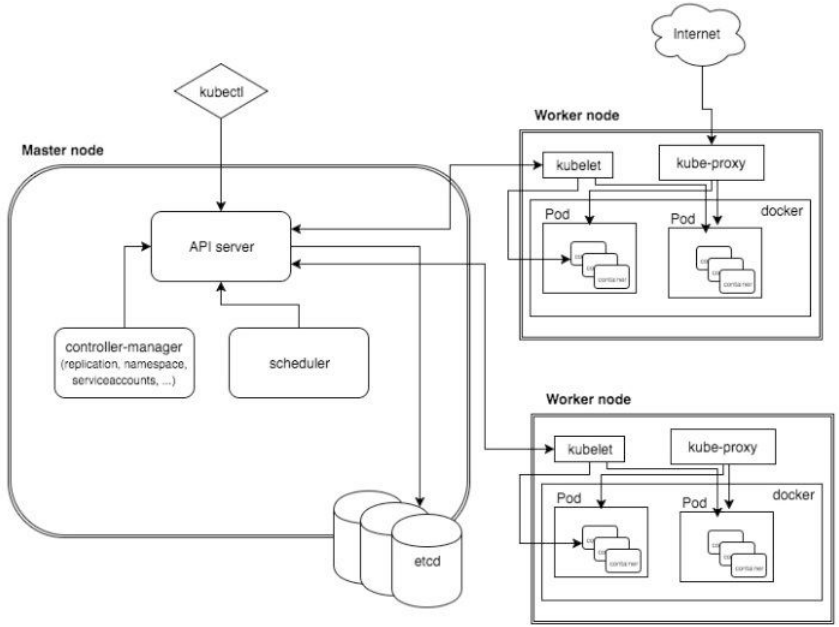
Kube Api Server

The Kubernetes API server validates and configures data for the api objects which include pods, services, replicationcontrollers, and others. The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact.



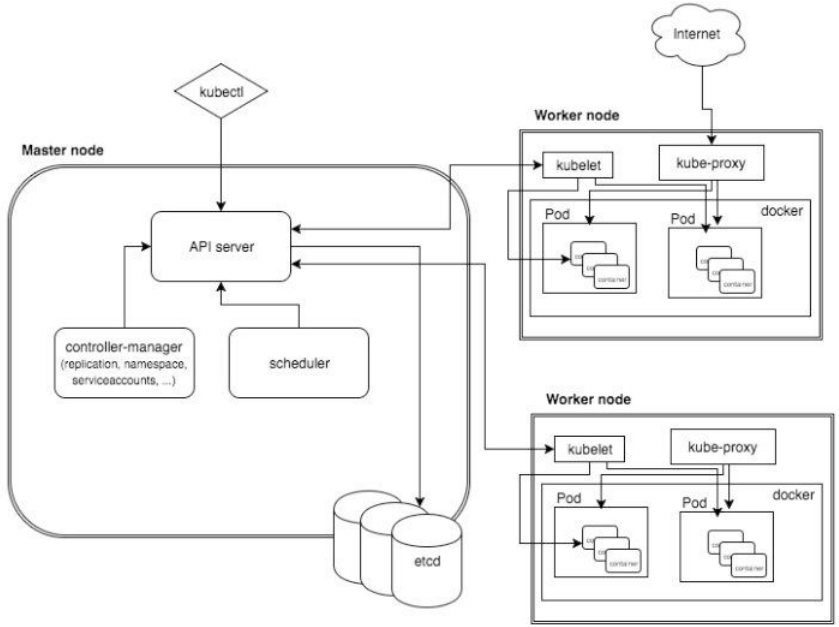
Kube Controller Manager

The Kubernetes Controller Manager is responsible for monitoring and managing the health of a cluster's services and replication controllers. Is a control loop that watches the shared state of the cluster through the apiserver and makes changes attempting to move the current state towards the desired state. Examples of controllers that ship with Kubernetes today are the replication controller, endpoints controller, namespace controller, and service accounts controller.



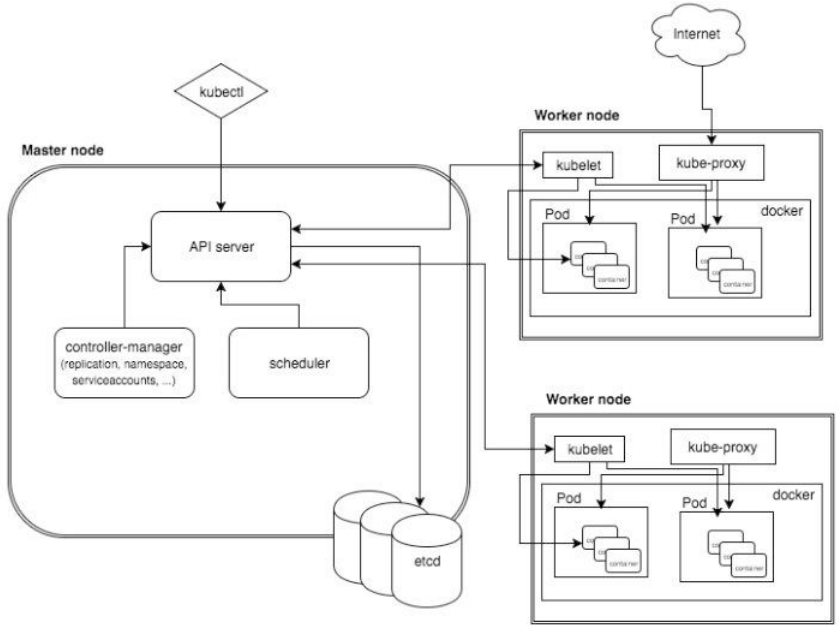
Kube Proxy

The Kubernetes network proxy runs on each node. This reflects services as defined in the Kubernetes API on each node and can do simple TCP, UDP, and SCTP stream forwarding or round robin TCP, UDP, and SCTP forwarding across a set of backends.



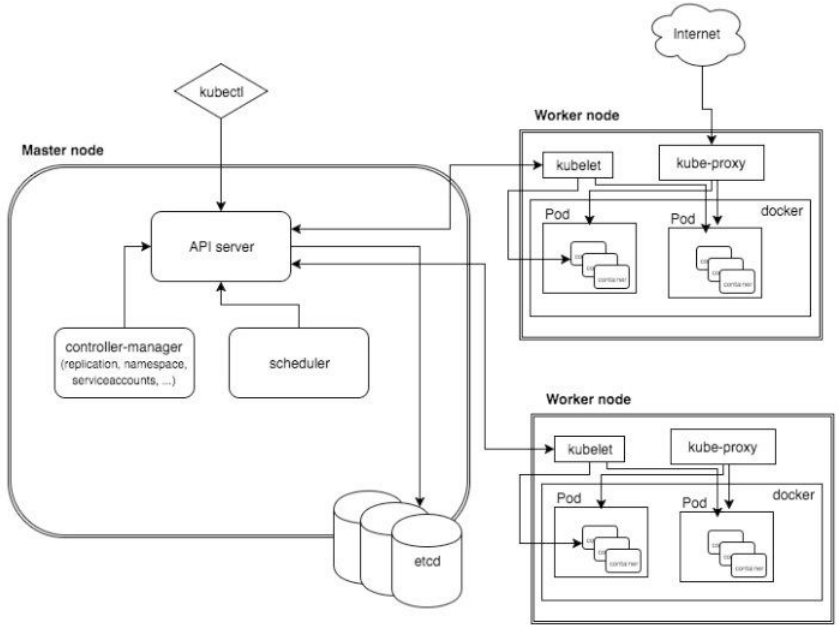
Kube Scheduler

The Kubernetes scheduler is responsible for assigning pods to nodes. It is a simple round-robin scheduler that assigns pods to nodes in a round-robin fashion. The scheduler determines which Nodes are valid placements for each Pod in the scheduling queue according to constraints and available resources.



Etcd

A distributed key-value store that is used to store the state of the cluster. It is only accessible from the API server for security reasons. etcd enables notifications to the cluster about configuration changes with the help of watchers. Notifications are API requests on each etcd cluster node to trigger the update of information in the node's storage.



Objects

Kubernetes objects are persistent entities in the Kubernetes system. Kubernetes uses these entities to represent the state of your cluster. Specifically, they can describe:

- What containerized applications are running (and on which nodes)
- The resources available to those applications
- The policies around how those applications behave, such as restart policies, upgrades, and fault-tolerance

Namespace

A Kubernetes namespace is a collection of objects. It is used to organize your cluster. If you delete a namespace, all objects in that namespace are deleted. You also need to reference the namespace in your commands, if needed.

Pod

Pods are the containers that run on the nodes of your cluster. They are the smallest unit of work in Kubernetes. A Pod is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers. A Pod's contents are always co-located and co-scheduled, and run in a shared context.

Replica Set

A Replica Set purpose is to maintain a stable set of replica Pods running at any given time. As such, it is often used to guarantee the availability of a specified number of identical Pods.

Deployment

A Deployment provides declarative updates for Pods and Replica Sets.

You describe a desired state in a Deployment, and the Deployment Controller changes the actual state to the desired state at a controlled rate. You can define Deployments to create new Replica Sets, or to remove existing Deployments and adopt all their resources with new Deployments.

Service

In Kubernetes, a Service is an abstraction which defines a logical set of Pods and a policy by which to access them (sometimes this pattern is called a micro-service). A Service is a collection of Pods that share a common purpose.

Interactions

There are multiple ways to do that. You can use a programmatic API, or you can use the command line tools. We have for example the Go client. In which you define an object that represents a Kubernetes cluster, and then you can use the API to interact with it.

But the most convenient way is to use the command line tool.

Get

The get subcommand is used to interact with objects in the cluster. For example, we can get a list of all the pods in the cluster.

```
kubectl get pods --all-namespaces
```

We can also get information about other objects, like deployments, namespaces, services and so on.

Apply

The apply subcommand is used to apply a manifest to the cluster. For example, we can apply a new deployment. Or modify an existing one. If the resource does not exist, it will be created. If it does exist, it will be updated.

kubectl apply -f manifest.yaml

Describe

The describe subcommand is used to describe objects in the cluster. For example, we can describe a deployment.

```
kubectl describe deployment my-deployment -n
my-namespace
```

Logs

The logs subcommand is used to get logs from pods, deployments, jobs and many other objects. For example, we can get the logs of a pod.

kubectl logs nginx -n my-namespace

Port-Forward

The port-forward subcommand is used to forward ports from a pod to the local machine. For example, we can forward the port 8080 of a pod to the local machine.

kubectl port-forward pod/mypod -n my-namespace 8080 8080

Exec

The exec subcommand is used to execute a command in a container in a pod. For example, we can execute a command in a pod.

kubectl exec mypod -- bash

Demo

Let’s create our first application in Kubernetes

Exercises

1. Multi Container Pod
2. Create a Deployment
3. Create a Service + Deployment
4. Interact with Services

Any Questions?