Kubernetes

An Overview

k21academy.com

What is a container?

- A container is like a portable package for your software. It wraps up everything your application needs to run—like the code, libraries, and any dependencies—into a single unit.
- You can think of a container as a lightweight, self-contained version of a virtual machine, but it's more efficient because it doesn't need to include a full operating system

Why Kubernetes?

- Let's say you need to run 50 different apps, each in its own container, so you have 50 containers.
- To keep these apps available all the time, you make two copies of each one, adding up to 100 containers.
- Now, you're responsible for 100 containers.

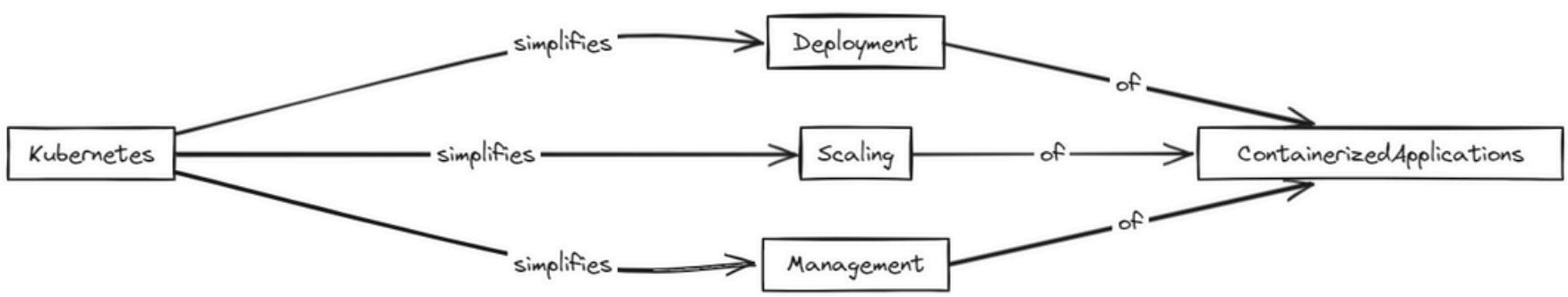
- Handling 100 containers by yourself? That sounds pretty tough.
- It would definitely be a challenging task to manage all of them alone

Orchestration

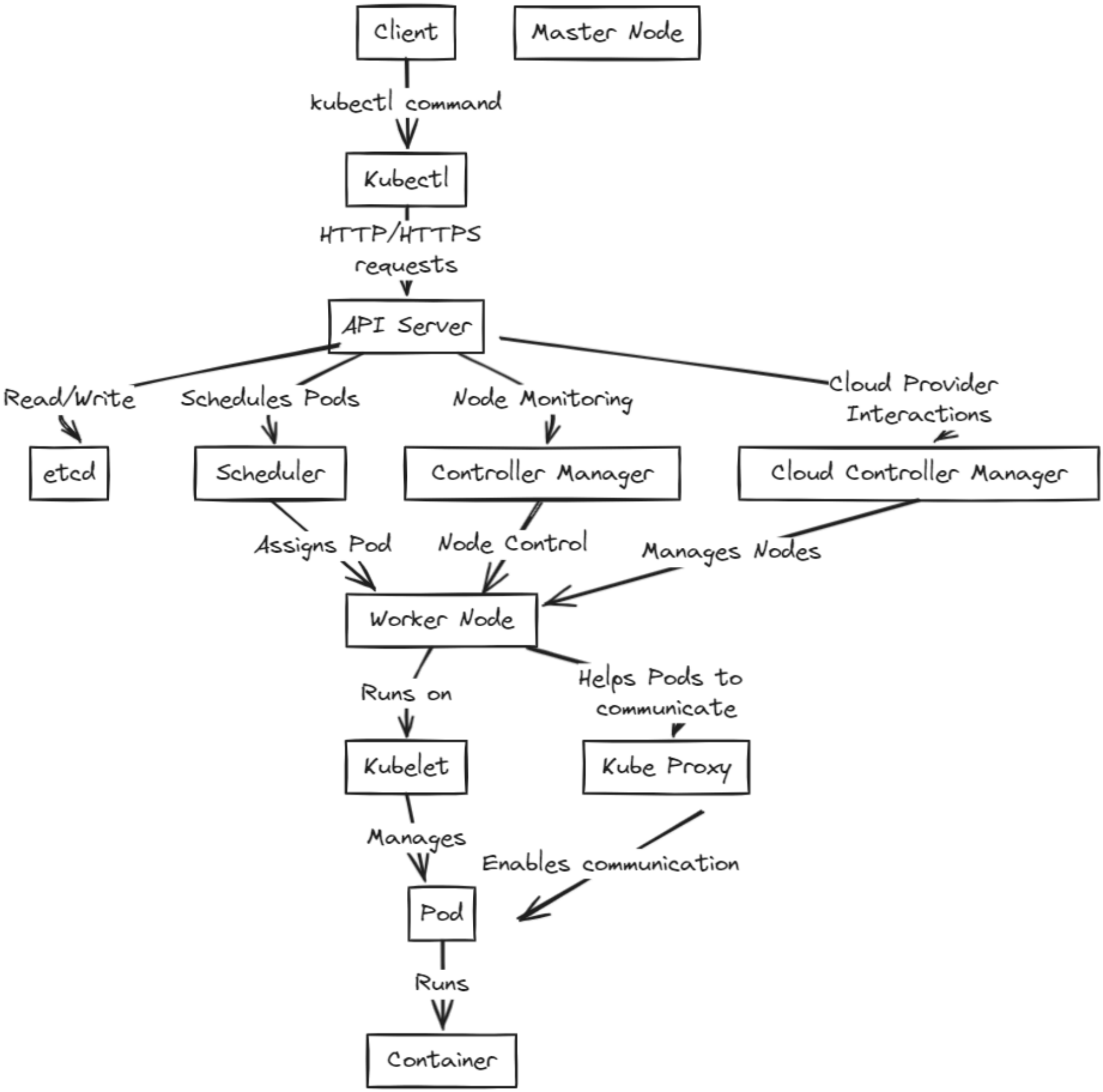
- A container orchestration tool can simplify managing lots of containers by automating their setup and upkeep.
- Kubernetes is a popular tool that does exactly this, making it easier to handle large numbers of containers.

What is Kubernetes?

Kubernetes, an open-source container orchestration platform, simplifies the deployment, scaling, and management of containerized applications. It's the go-to solution for streamlining operations in modern IT environments.



Kubernetes Architecture



Control Plane (Master Node)

This is like the brain of the cluster. It makes decisions about what should be running and where. It keeps track of all the things you want to run and makes sure they're running the way you want them to. It's responsible for managing the cluster, scheduling applications, and making sure everything is working as it should.

Here are the key parts of a master node:

- API Server: The communication hub for all Kubernetes operations, processing commands and managing the state of the cluster.
- Scheduler: This decides which nodes are best suited to run your applications based on resources and other factors.
- Kubernetes Controller Manager: This component manages various controllers that help keep your applications running as you've defined.
- etcd: A secure, distributed database that Kubernetes uses to store all crucial data about your cluster.

Worker Nodes

Worker nodes are the servers in a Kubernetes system where your applications actually run.

Here are the key parts of a worker node:

- Kubelet: This is a program on each worker node that makes sure the containers are running as they should be according to the setup you've specified.
- Container Runtime: This is the software that runs the containers. Common examples include Docker and containerd.
- kube-proxy: This manages the network connections for the containers, helping your apps communicate both inside and outside the cluster.

Pods: The fundamental unit in Kubernetes, encapsulating one or more containers.

Service: This provides a stable way to access your containers, making it easier for applications to run reliably.

Namespace: Helps organize resources within your cluster, similar to folders on a computer.

Volumes: Support for various storage types, helping your applications keep data even if containers restart.

Secrets and ConfigMaps: These help manage sensitive information and configuration details safely and separately from your app's code.

Deployment: Manages the rollout and scaling of your applications, making updates smoother.

StatefulSets: Ideal for applications that need stable storage and unique network identifiers.

DaemonSets: Ensure that specific processes run on all or certain nodes within your cluster.

Jobs and CronJobs: Handle tasks that run once or on a schedule, useful for maintenance and batch processing.

Ingress: Manages external access to your services, typically useful for web traffic

Summary

Kubernetes operates on a "Master-Worker" cluster structure:

- Master Node: Handles essential processes required to manage the cluster.
- Worker Nodes: Execute your applications.

When using Kubernetes, you provide instructions on the desired state of your application using YAML or JSON configuration files. Kubernetes then works to achieve and maintain this state.

For example, if you want to deploy three different Spring Boot applications, each with two replicas and running on specific ports, you would:

• Create the necessary manifest files in YAML or JSON format.

• Submit these files to Kubernetes.

• Kubernetes automatically manages the deployment and maintenance of these applications based on the instructions provided. This setup ensures that the system operates as planned with very little need for manual oversight.