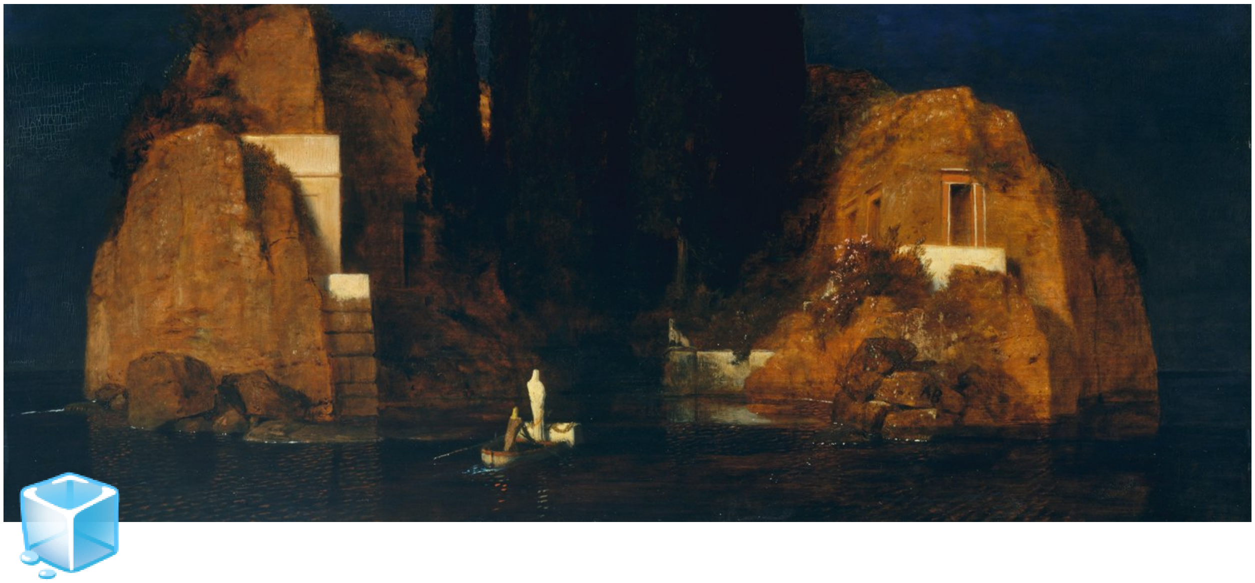
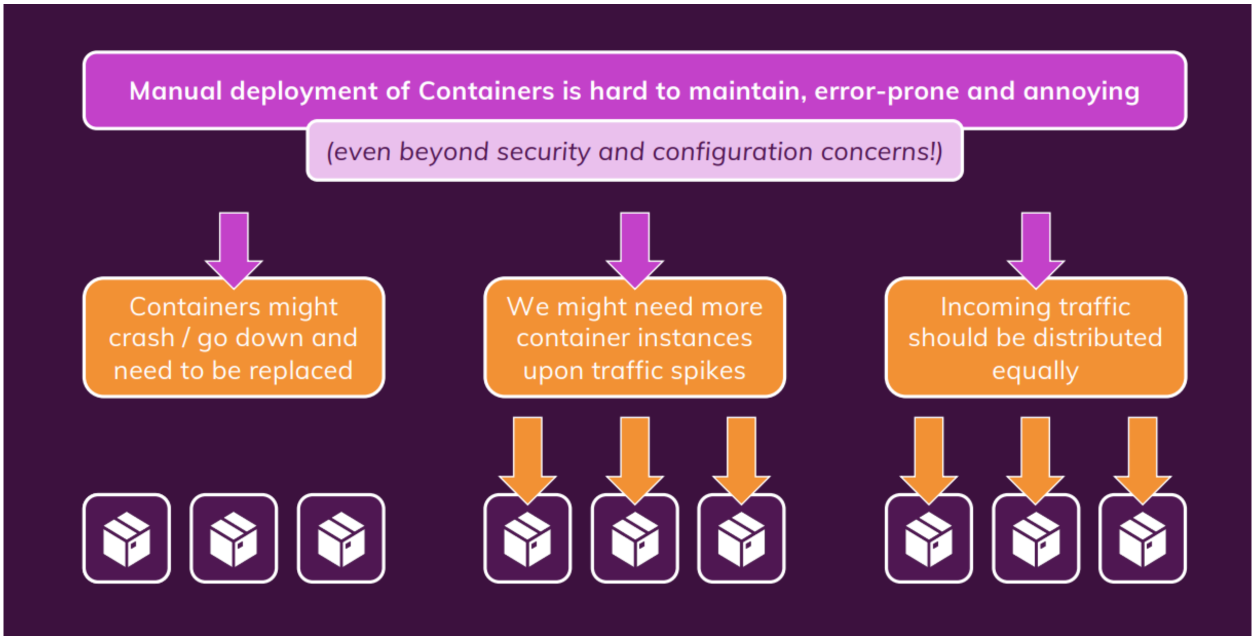


13 - Getting Started with Kubernetes

The Problem we currently have



Manual deployment of Containers is hard to maintain, error-prone and annoying

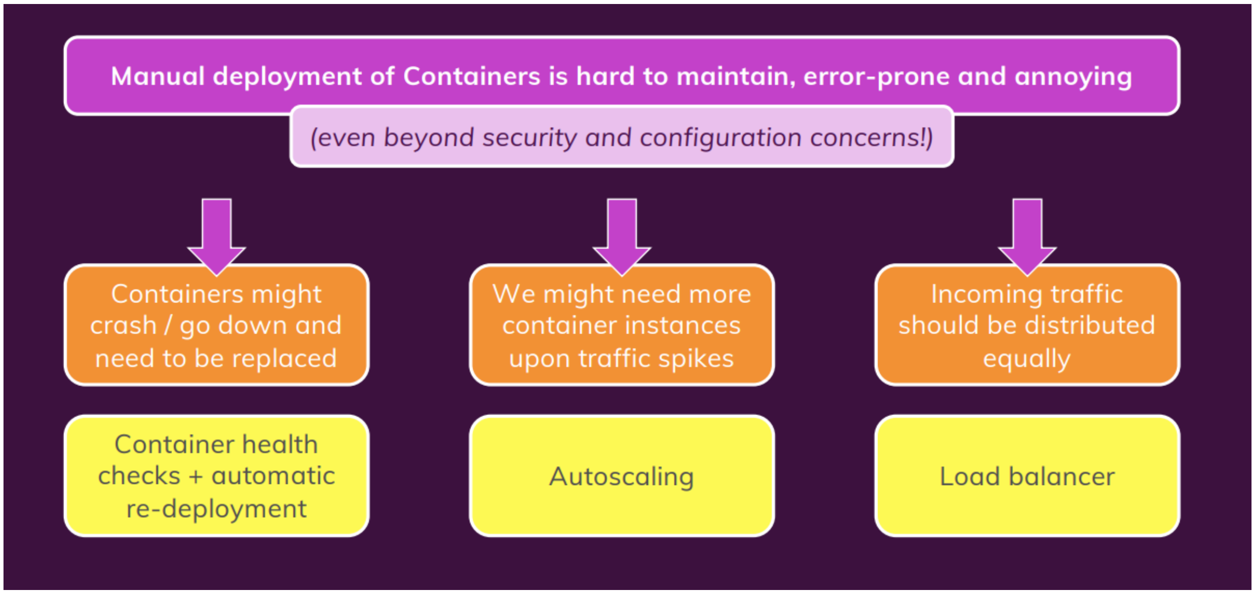
(even beyond security and configuration concerns!)

Containers might crash / go down and need to be replaced

We might need more container instances upon traffic spikes

Incoming traffic should be distributed equally

Some Services, like EWS already offer a solution to this problem



Manual deployment of Containers is hard to maintain, error-prone and annoying

(even beyond security and configuration concerns!)

Containers might crash / go down and need to be replaced

Container health checks + automatic re-deployment

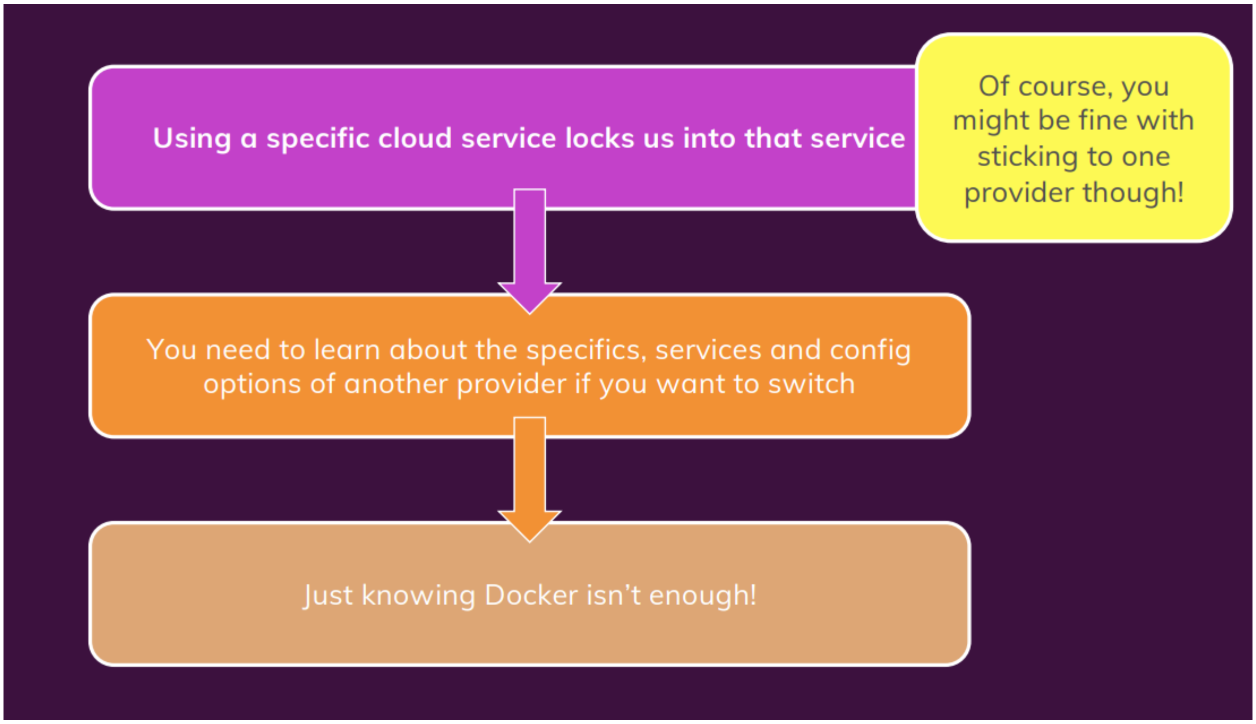
We might need more container instances upon traffic spikes

Autoscaling

Incoming traffic should be distributed equally

Load balancer

But that locks us in into a single Web Service and we have to use only their tools



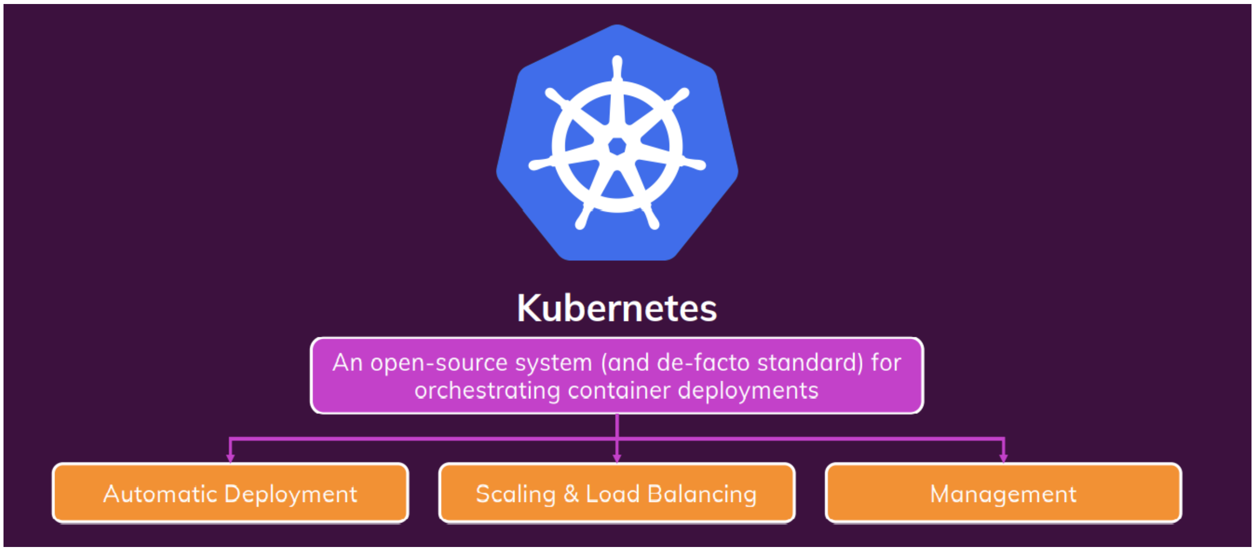
Using a specific cloud service locks us into that service

Of course, you might be fine with sticking to one provider though!

You need to learn about the specifics, services and config options of another provider if you want to switch

Just knowing Docker isn't enough!

What is Kubernetes



Kubernetes

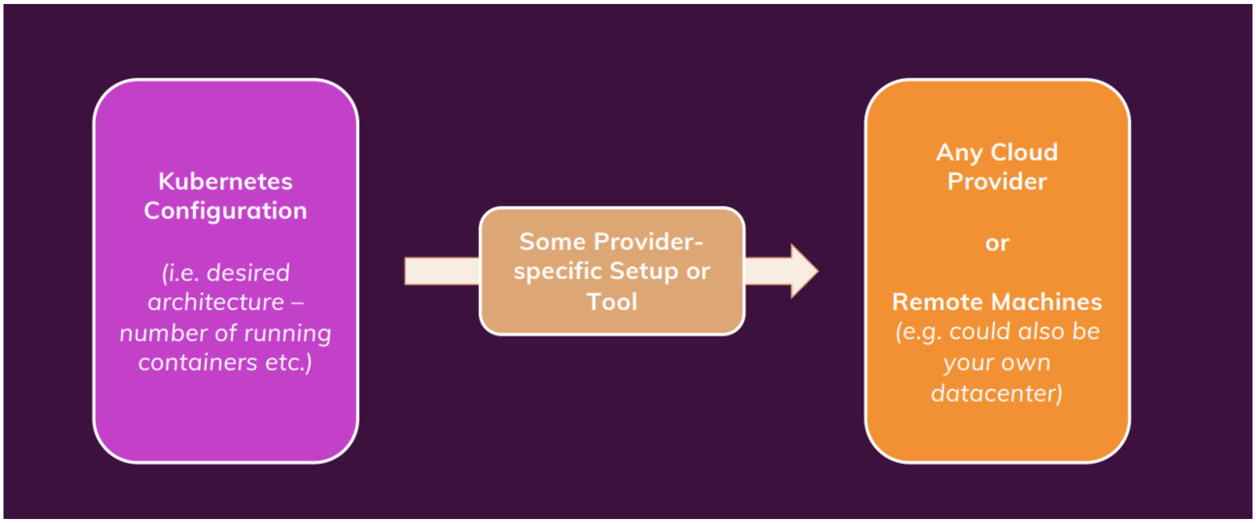
An open-source system (and de-facto standard) for orchestrating container deployments

Automatic Deployment

Scaling & Load Balancing

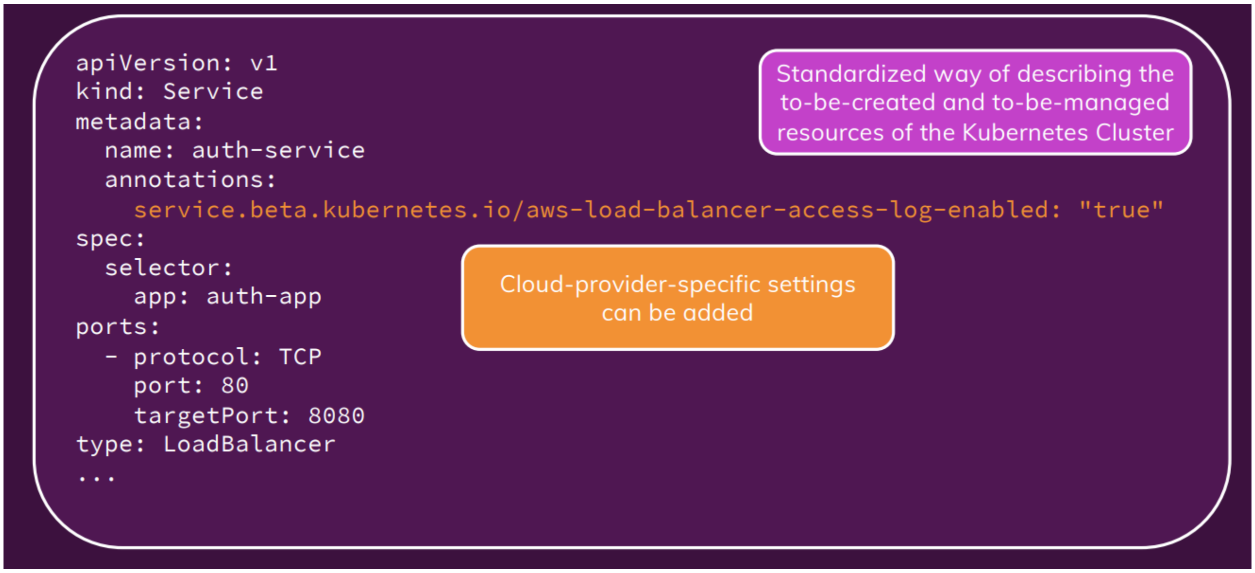
Management

💡 Kubernetes is like **Docker Compose** for **MULTIPLE machines**



Kubernetes Configuration
(i.e. desired architecture – number of running containers etc.)
Some Provider-specific Setup or Tool
Any Cloud Provider
or
Remote Machines
(e.g. could also be your own datacenter)

The Syntax

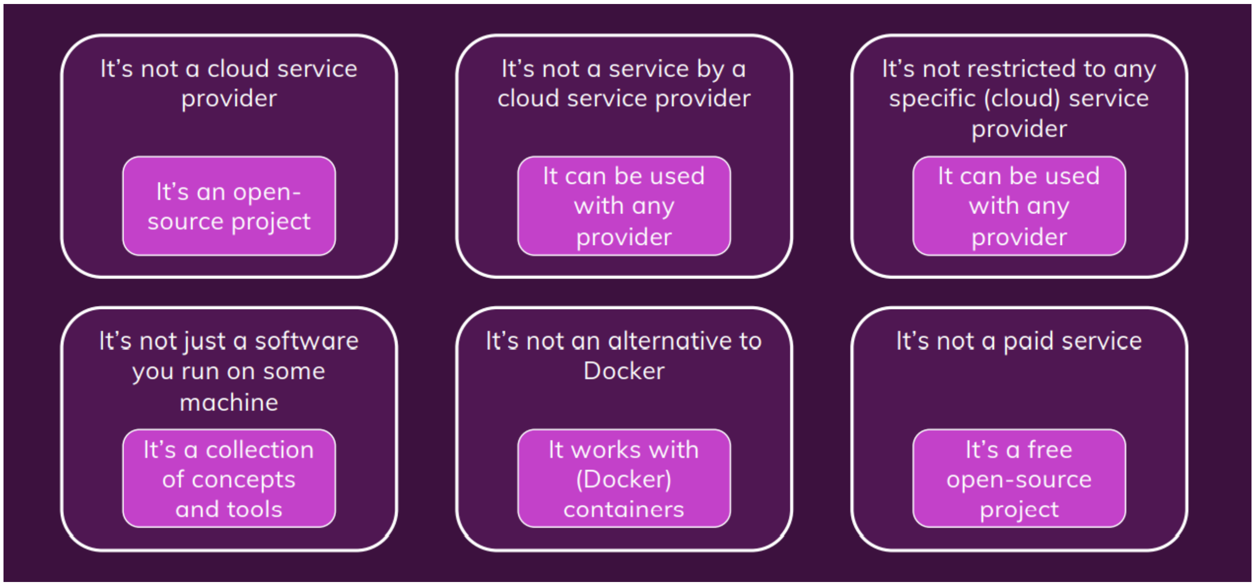


```
apiVersion: v1
kind: Service
metadata:
  name: auth-service
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-access-log-enabled: "true"
spec:
  selector:
    app: auth-app
ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
type: LoadBalancer
...
```

Standardized way of describing the to-be-created and to-be-managed resources of the Kubernetes Cluster

Cloud-provider-specific settings can be added

What Kubernetes is NOT



It's not a cloud service provider

It's an open-source project

It's not a service by a cloud service provider

It can be used with any provider

It's not restricted to any specific (cloud) service provider

It can be used with any provider

It's not just a software you run on some machine

It's a collection of concepts and tools

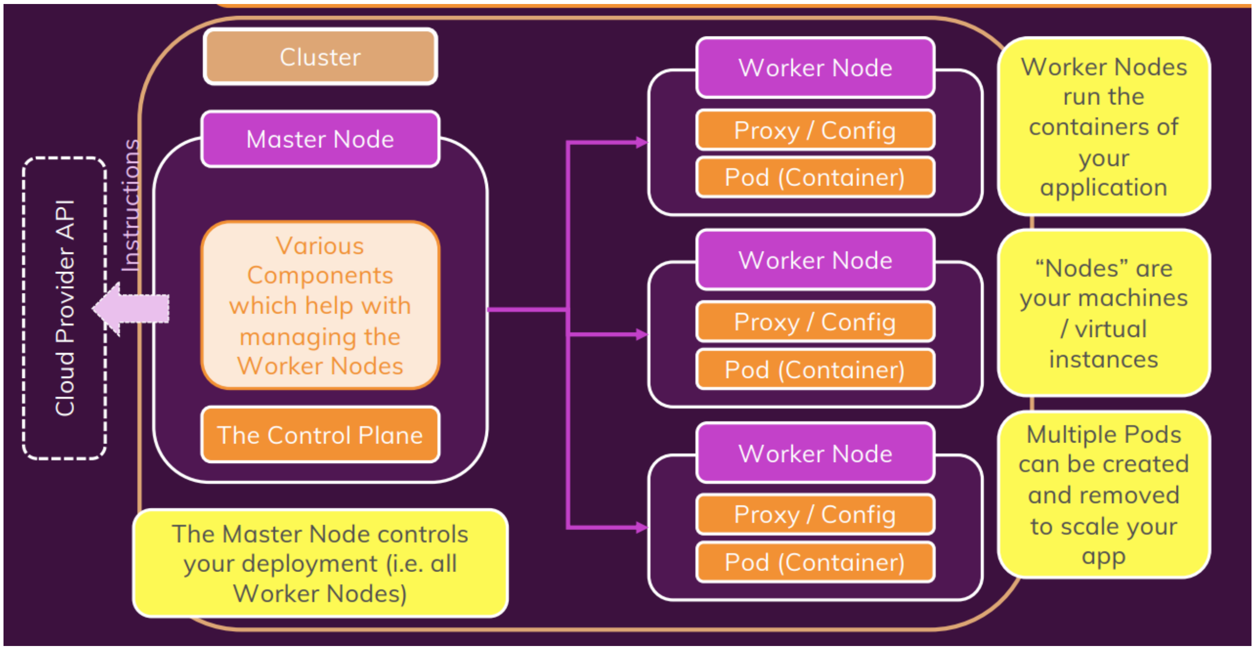
It's not an alternative to Docker

It works with (Docker) containers

It's not a paid service

It's a free open-source project

Kubernetes Architecture



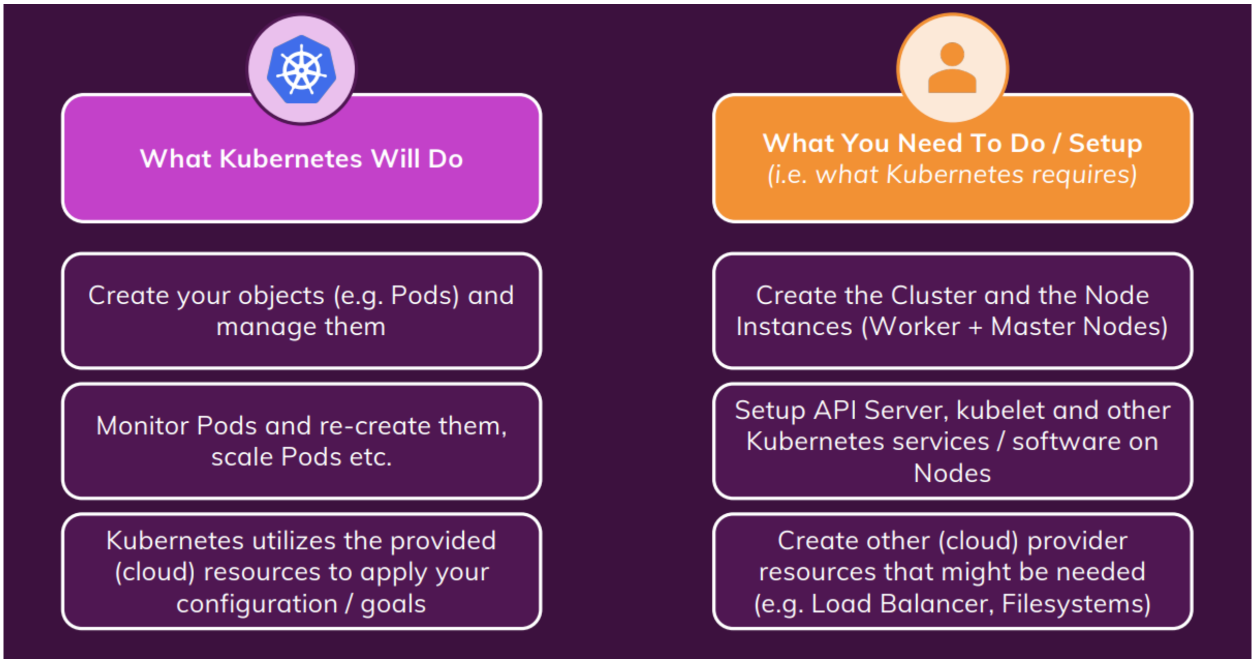
The Master Node controls your deployment (i.e. all Worker Nodes)

Worker Nodes run the containers of your application

"Nodes" are your machines / virtual instances

Multiple Pods can be created and removed to scale your app

The work you do vs the work Kubernetes does



What Kubernetes Will Do

Create your objects (e.g. Pods) and manage them

Monitor Pods and re-create them, scale Pods etc.

Kubernetes utilizes the provided (cloud) resources to apply your configuration / goals

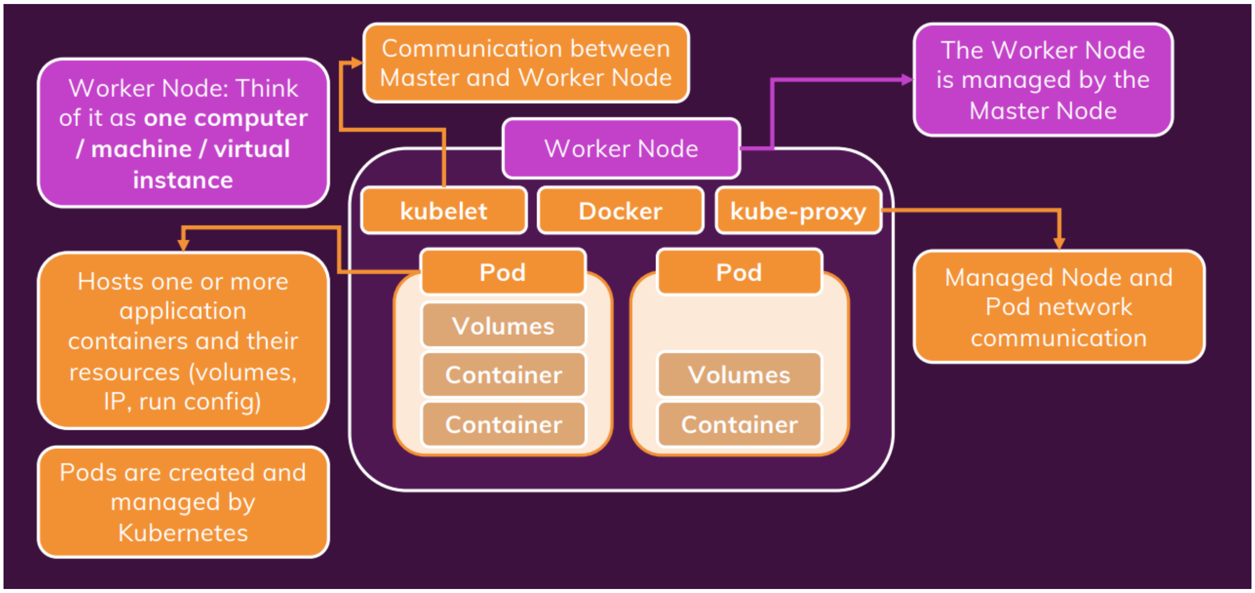
What You Need To Do / Setup
(i.e. what Kubernetes requires)

Create the Cluster and the Node Instances (Worker + Master Nodes)

Setup API Server, kubelet and other Kubernetes services / software on Nodes

Create other (cloud) provider resources that might be needed (e.g. Load Balancer, Filesystems)

A closer look at the Worker Nodes



Worker Node: Think of it as **one computer / machine / virtual instance**

Communication between Master and Worker Node

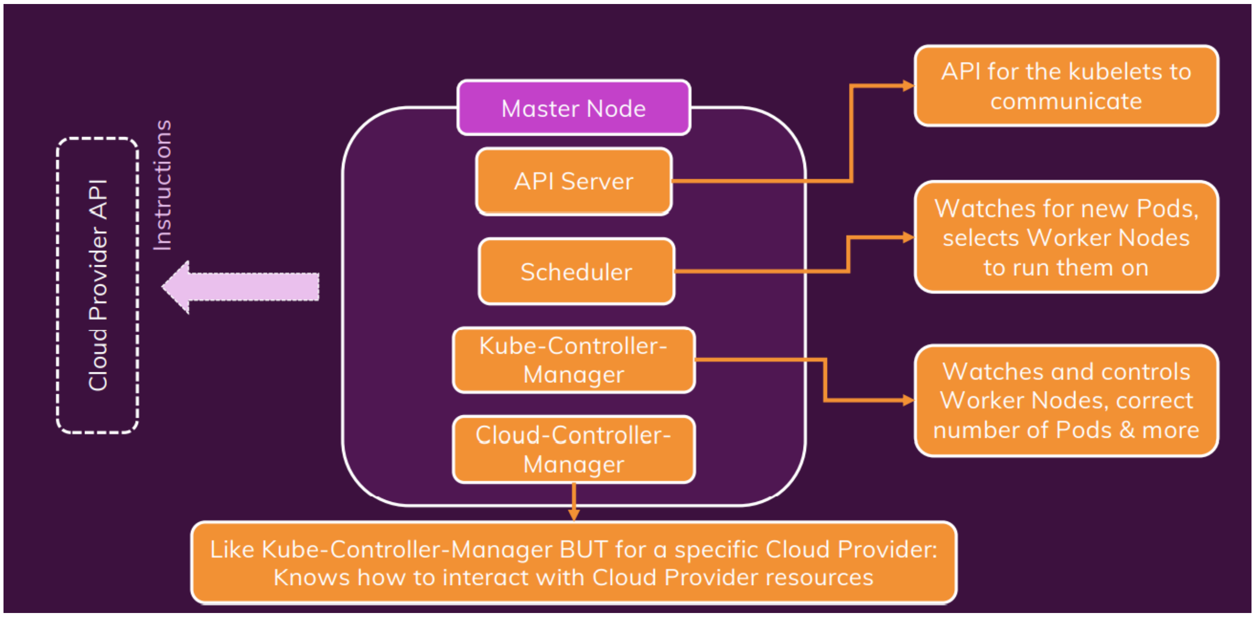
The Worker Node is managed by the Master Node

Hosts one or more application containers and their resources (volumes, IP, run config)

Pods are created and managed by Kubernetes

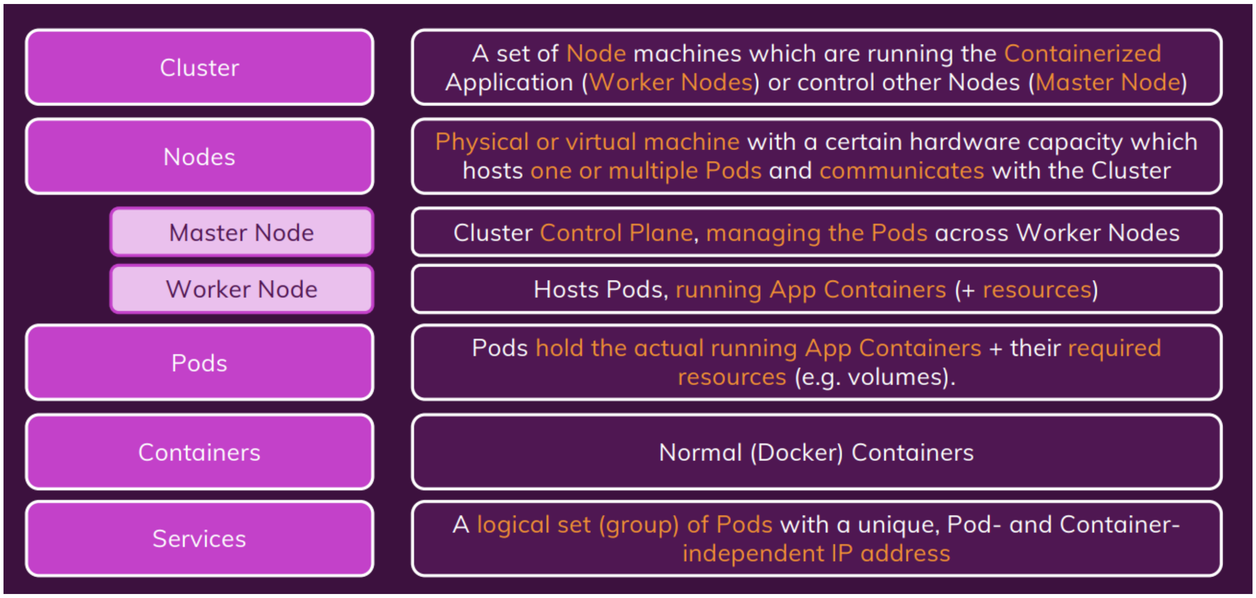
Managed Node and Pod network communication

A closer look at Master Nodes



Master Node
API Server
Scheduler
Kube-Controller-Manager
Cloud-Controller-Manager
API for the kubelets to communicate
Watches for new Pods, selects Worker Nodes to run them on
Watches and controls Worker Nodes, correct number of Pods & more
Like Kube-Controller-Manager BUT for a specific Cloud Provider: Knows how to interact with Cloud Provider resources
Instructions
Cloud Provider API

Summary and Core Concepts



Cluster	A set of Node machines which are running the Containerized Application (Worker Nodes) or control other Nodes (Master Node)
Nodes	Physical or virtual machine with a certain hardware capacity which hosts one or multiple Pods and communicates with the Cluster
Master Node	Cluster Control Plane, managing the Pods across Worker Nodes
Worker Node	Hosts Pods, running App Containers (+ resources)
Pods	Pods hold the actual running App Containers + their required resources (e.g. volumes).
Containers	Normal (Docker) Containers
Services	A logical set (group) of Pods with a unique, Pod- and Container-independent IP address