

## We (Maybe) Have A 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

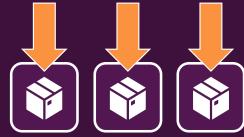






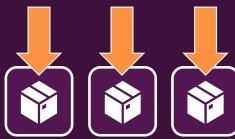
might need to replace/restart the container.

We might need more container instances upon traffic spikes



containers.

Incoming traffic should be distributed equally



container is doing all the work.



## Services Like AWS ECS Can Help!

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

auto restart containers when they crash

We might need more container instances upon traffic spikes

Autoscaling

Incoming traffic should be distributed equally

Load balancer



### **But That Locks Us In!**

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!



### **Kubernetes To The Rescue**



#### **Kubernetes**

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

Automatic Deployment

Scaling & Load Balancing

Management



## **Using Kubernetes**

# Kubernetes is like Docker-Compose for multiple machines



# Why Kubernetes?

Kubernetes Configuration

(i.e. desired architecture – number of running containers etc.) Some Providerspecific Setup or Tool Any Cloud Provider

or

Remote Machines (e.g. could also be your own datacenter)



## Extensible, Yet Standardized Configuration

```
apiVersion: v1
                                                     Standardized way of describing the
kind: Service
                                                     to-be-created and to-be-managed
metadata:
                                                     resources of the Kubernetes Cluster
  name: auth-service
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-access-log-enabled: "true"
spec:
  selector:
                                Cloud-provider-specific settings
    app: auth-app
                                        can be added
ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
type: LoadBalancer
```



#### What Kubernetes IS and IS NOT

It's not a cloud service provider

It's an opensource 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

**Core Kubernetes Concepts & Architecture** ACADE MIND Cluster Worker Node Worker Nodes run the containers of Proxy / Config Master Node Instructions your Pod (Container) application Cloud Provider API Various Worker Node "Nodes" are Components your machines which help with Proxy / Config / virtual managing the instances Worker Nodes Pod (Container) The Control Plane Multiple Pods Worker Node can be created and removed Proxy / Config to scale your The Master Node controls app Pod (Container) your deployment (i.e. all Worker Nodes)



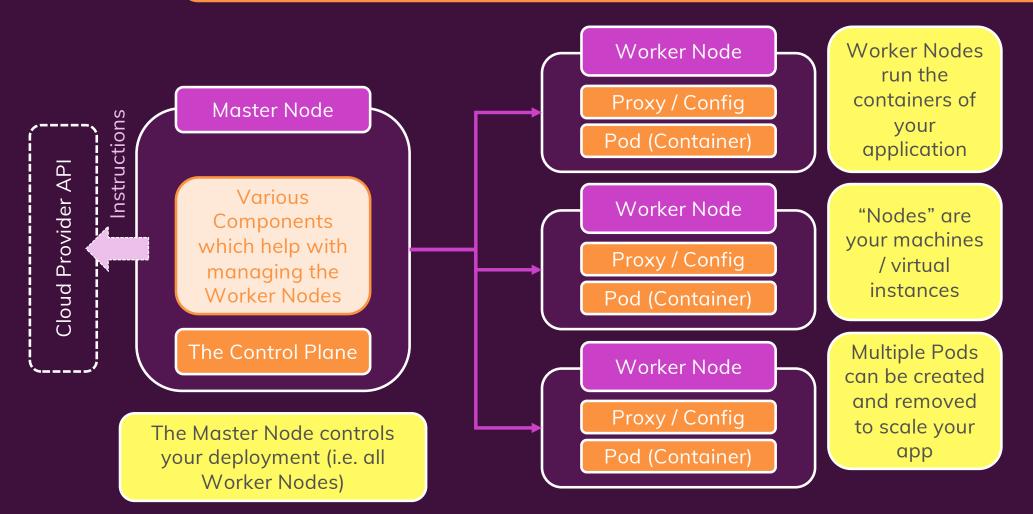
Kubernetes

#### A Closer Look At The Worker Nodes

Communication between The Worker Node Master and Worker Node is managed by the Worker Node: Think Master Node of it as **one computer** / machine / virtual Worker Node instance kubelet Docker kube-proxy Pod Pod Managed Node and Hosts one or more Pod network application Volumes communication containers and their Volumes resources (volumes, Container IP, run config) Container Container Pods are created and managed by

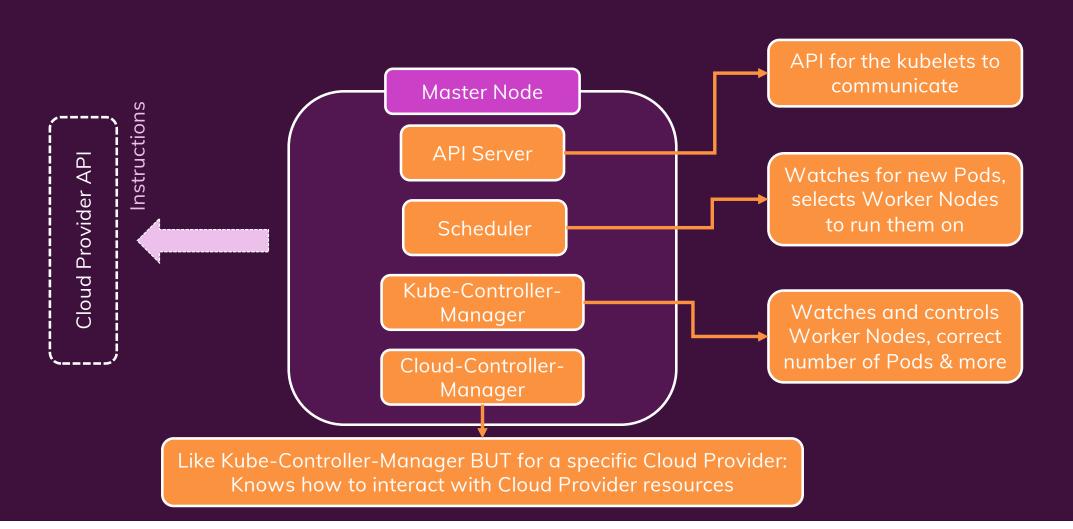


## **Core Kubernetes Concepts & Architecture**





# Core Kubernetes Concepts & Architecture





## Your Work / Kubernetes' Work



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)



#### **Core Components**

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