

Kubernetes

-
- Understanding need of Kubernetes
 - What is Kubernetes?
 - Core Concepts & Architecture
 - Kubernetes Cluster – Master & Worker nodes
 - Kubernetes Setup
 - Understanding Kubernetes Objects

We (May) Have A Problem

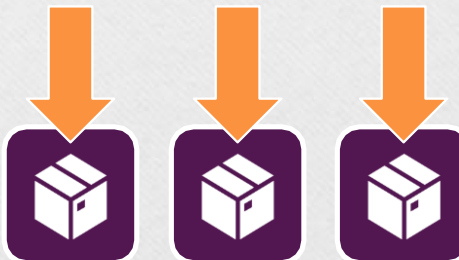
Manual deployment of Containers is hard to maintain, error-prone and annoying

(even beyond security and configuration concerns!)

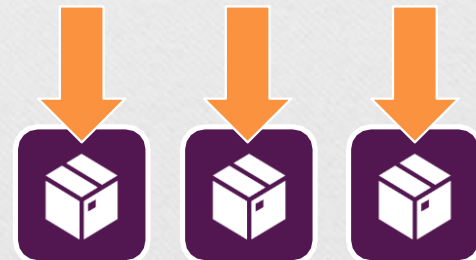
Containers might
crash /go down and
need to be replaced



We might need more
container instances
upon traffic spikes



Incoming traffic
should be distributed
equally



Services Like AWS ECS / Azure AKS 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

↓

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

What Kubernetes IS and 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

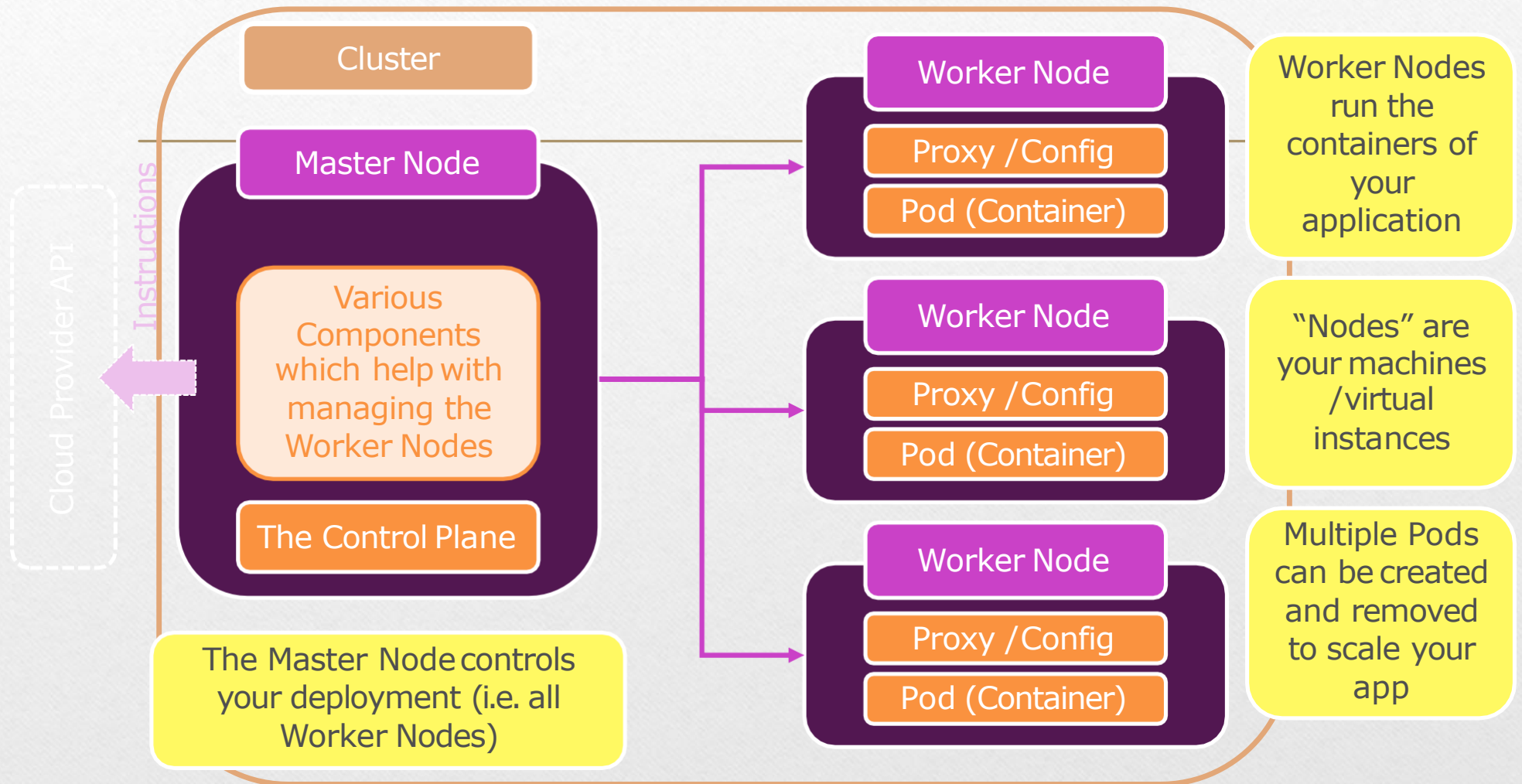
It's not an alternative to Docker

It works with (Docker) containers

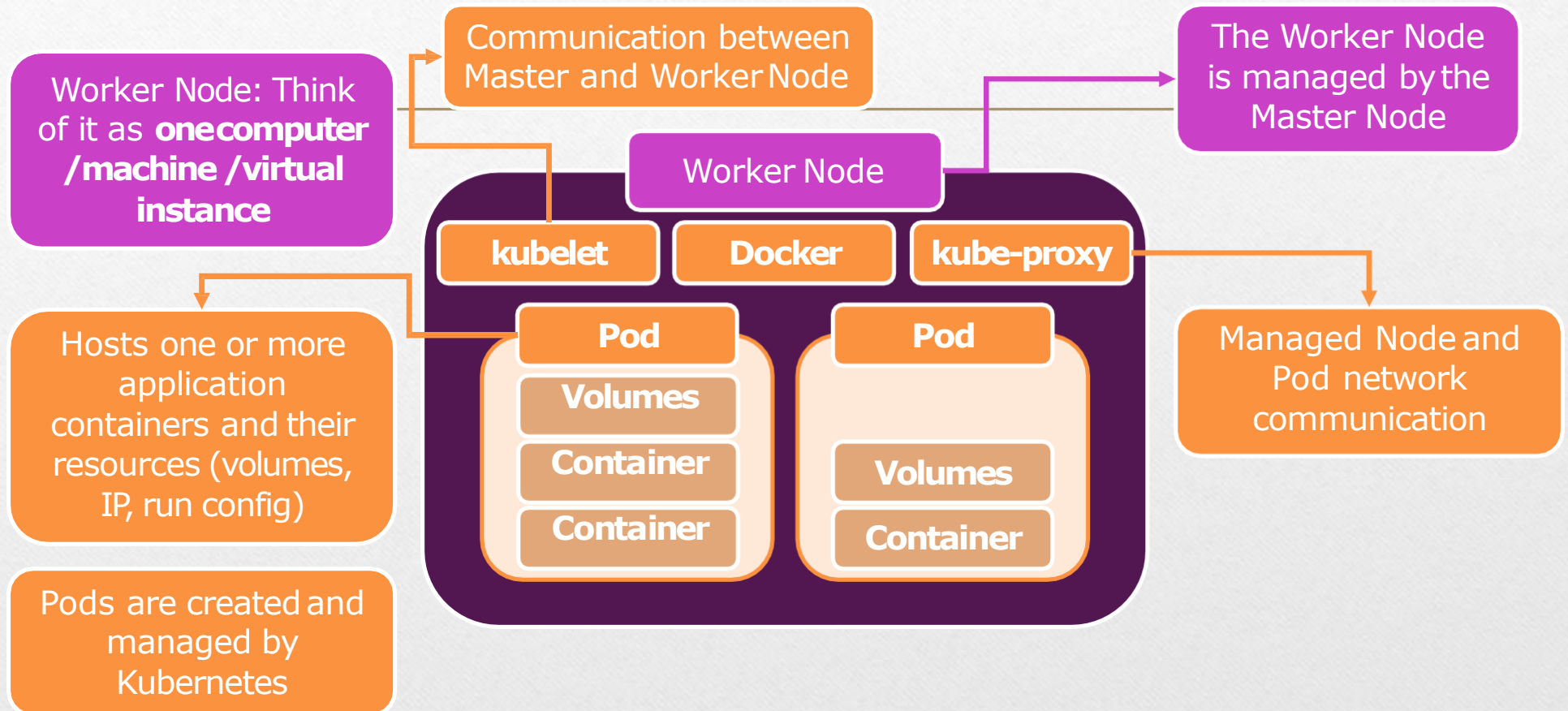
It's not a paid service

It's a free open-source project

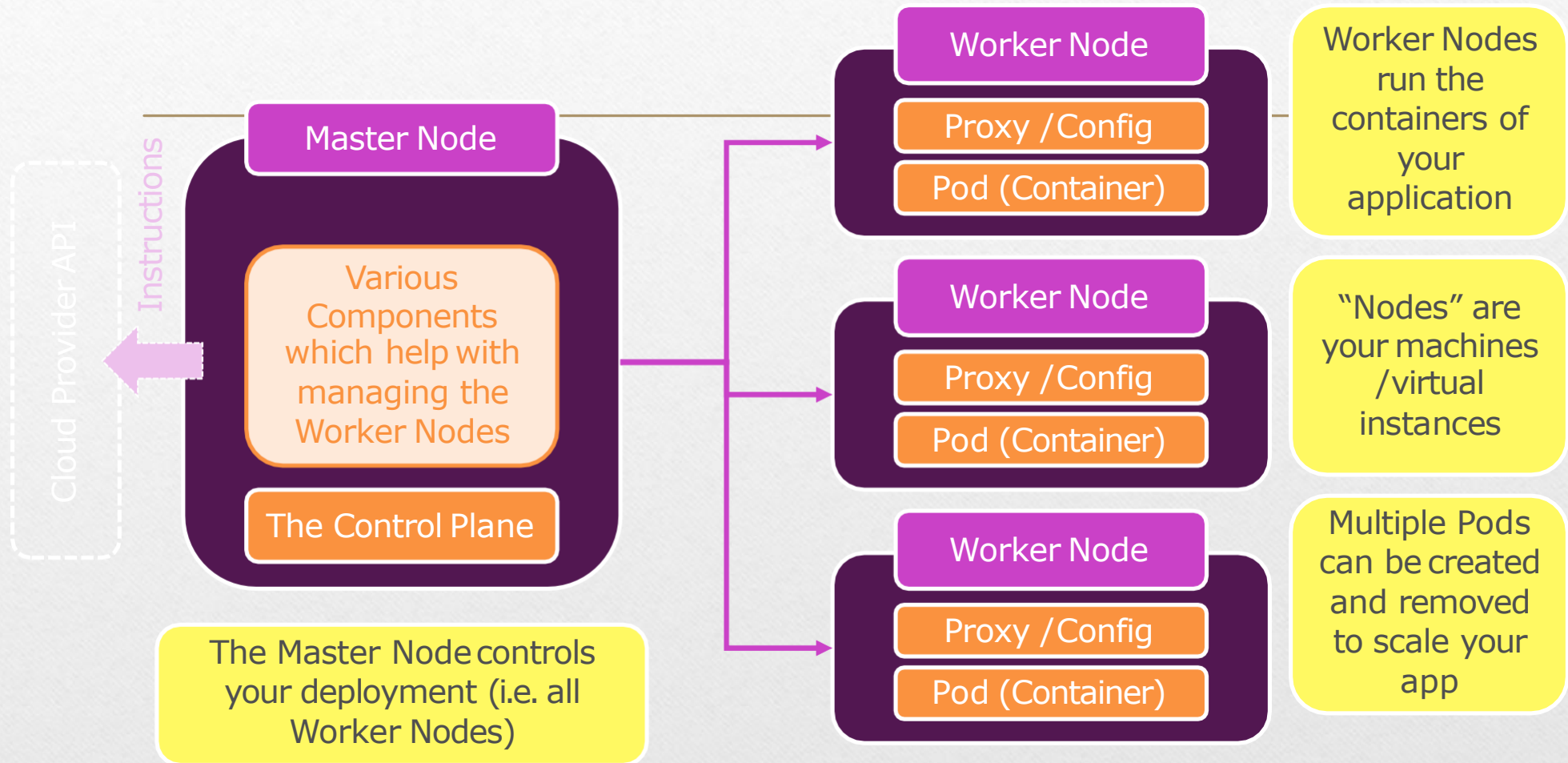
Core Kubernetes Concepts & Architecture



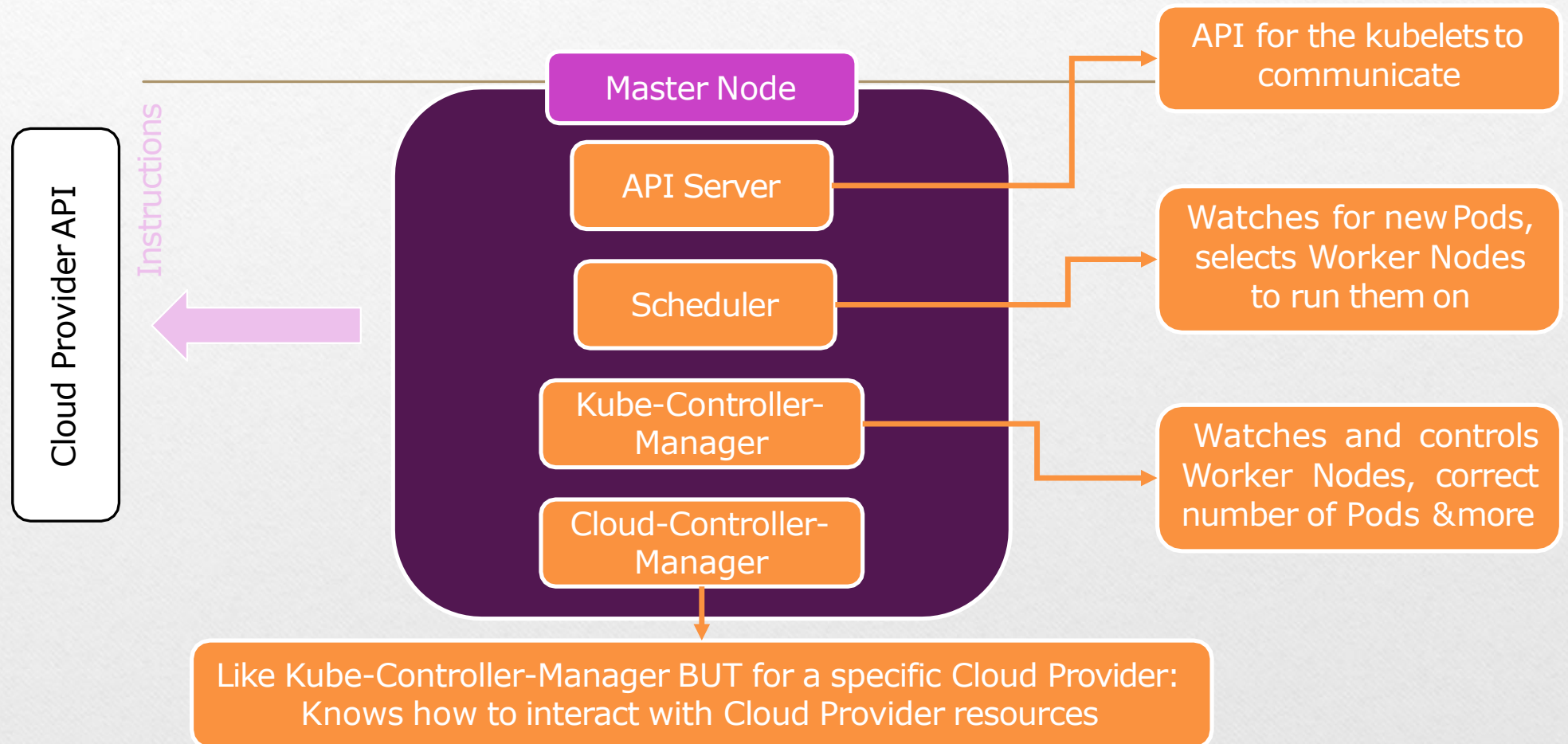
A Closer Look At The Worker Nodes



Core Kubernetes Concepts & Architecture



Core Kubernetes Concepts & Architecture





What Kubernetes Will Do

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

Monitor Pods and re-createthem, 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).

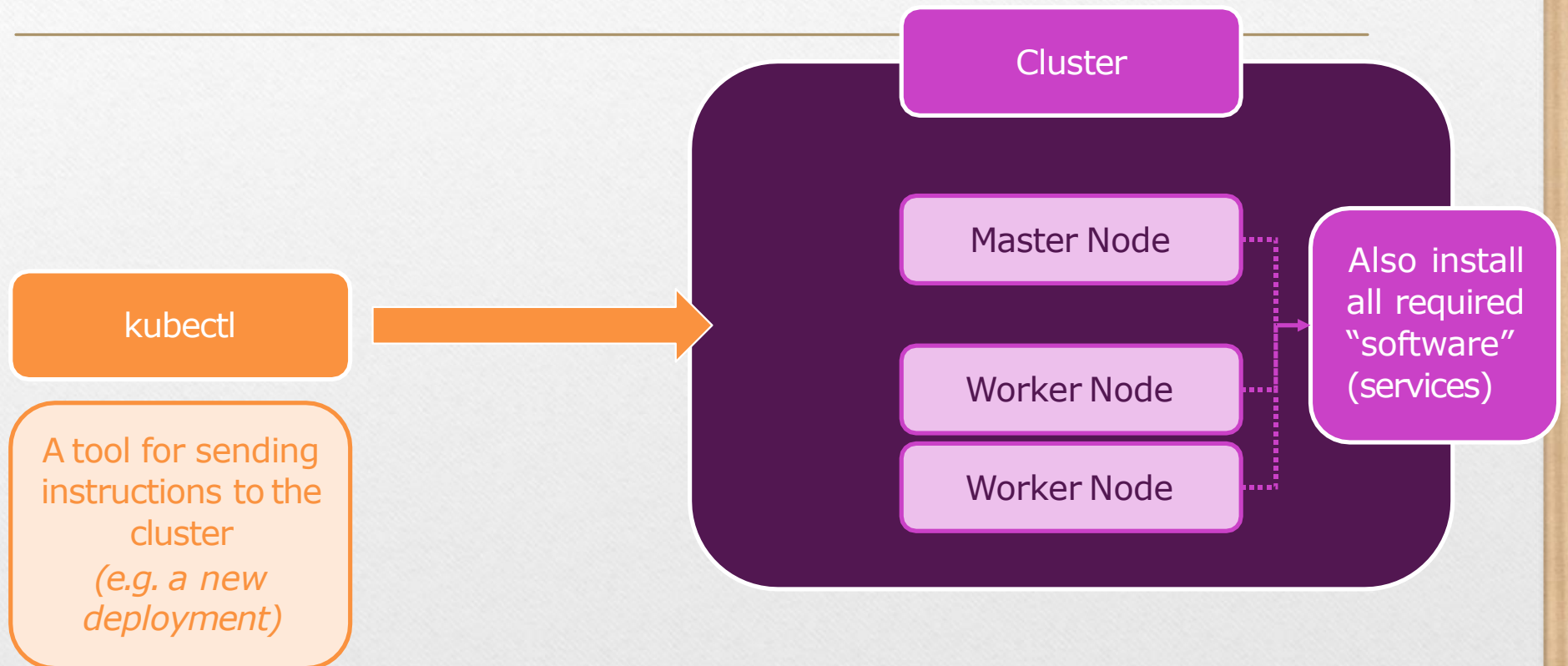
Containers

Normal (Docker) Containers

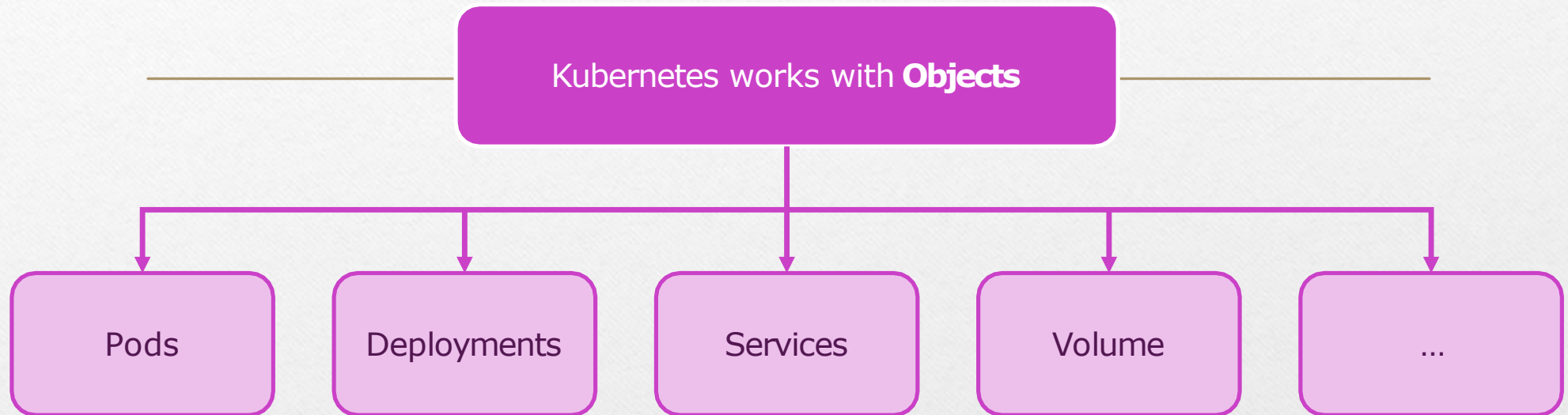
Services

A **logical set (group)** of **Pods** with a unique, Pod- and Container-independent **IP address**

Installation



Understanding Kubernetes Objects



Objects can be created in two ways:
Imperatively or Declaratively

The ‘Pod’ Object



The smallest “unit” Kubernetes interacts with

Contains and runs one or multiple containers

The most common use-case is “one container per Pod”

Pods contain shared resources (e.g. volumes) for all Pod containers

Has a cluster-internal IP by default

Containers inside a Pod can communicate via localhost

Pods are designed to be **ephemeral**: Kubernetes will start, stop and replace them as needed.

For Pods to be managed for you, you need a “**Controller**” (e.g. a “Deployment”)

The "Deployment" Object



Controls (multiple) Pods

You set a desired state,
Kubernetes then changes
the actual state

Define which Pods and
containers to run and the
number of instances

Deployments can be
paused, deleted and
rolled back

Deployments can be
scaled dynamically (and
automatically)

You can change the
number of desired Pods
as needed

Deployments manage a Pod for you, you can also create multiple Deployments

You therefore typically don't directly control Pods, instead you use
Deployments to set up the desired endstate

The "Service" Object



Exposes Pods to the Cluster or Externally

Pods have an internal IP by default – it changes when a Pod is replaced

Finding Pods is hard if the IP changes all the time

Services group Pods with a shared IP

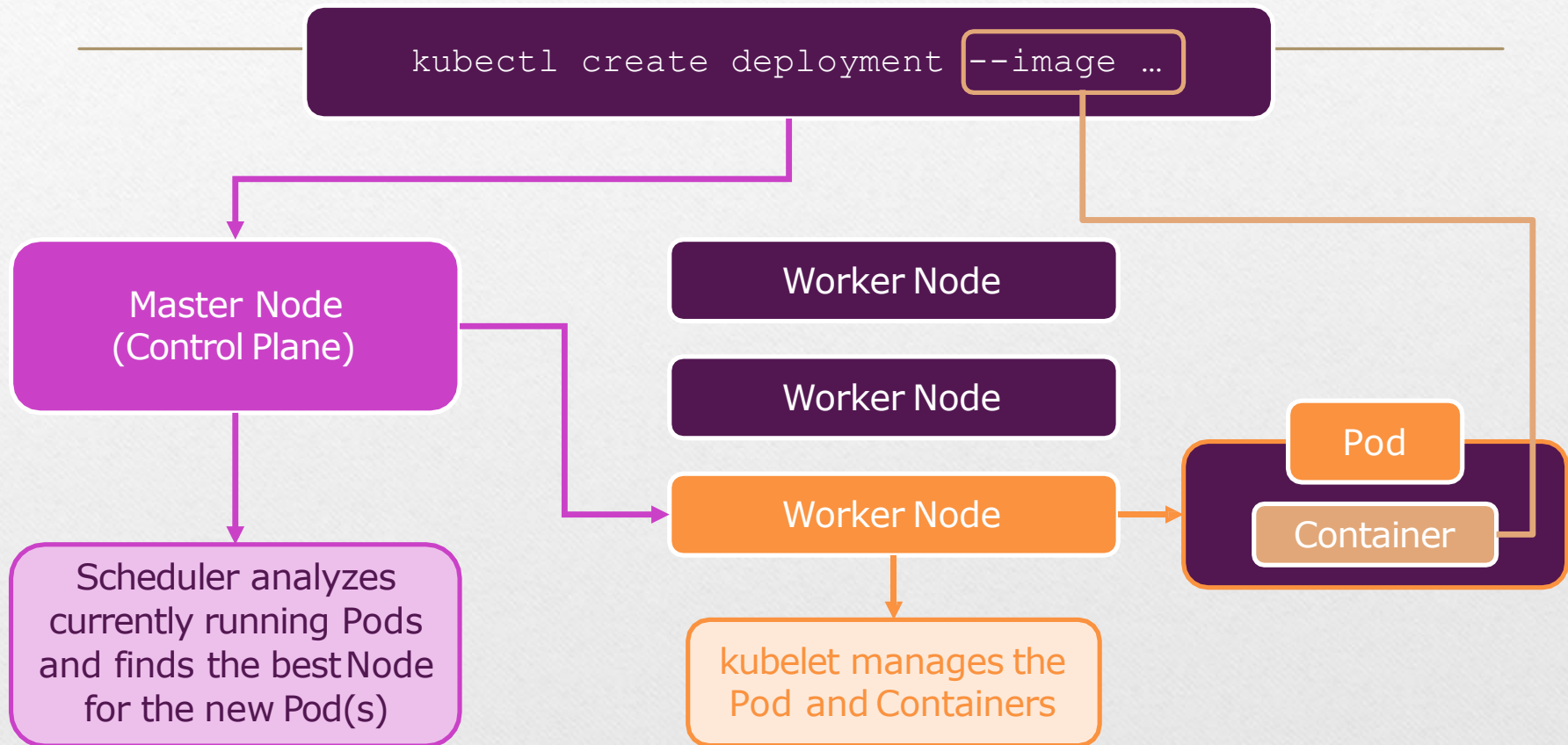
Services can allow external access to Pods

The default (internal only) can be overwritten

Without Services, Pods are very hard to reach and communication is difficult

Reaching a Pod from outside the Cluster is not possible at all without Services

Behind The Scenes



Configuring Kubernetes

Imperative

```
kubectl create deployment ...
```

Individual commands are executed to trigger certain Kubernetes actions

Comparable to using **docker run** only

Declarative

```
kubectl apply -f config.yaml
```

A config file is defined and applied to change the desired state

Comparable to using **Docker Compose** with compose files

A Resource Definition

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: second-app
spec:
  selector:
    matchLabels:
      app: second-dummy
  replicas: 1
  template:
    metadata:
      labels:
        app: second-dummy
    spec:
      containers:
      - name: second-node
        image: "academind/kub-first-app"
```


Thank You
