Modern Software Architecture, Containerization, and Kubernetes

INTRODUCTION TO KUBERNETES

Frank Heilmann
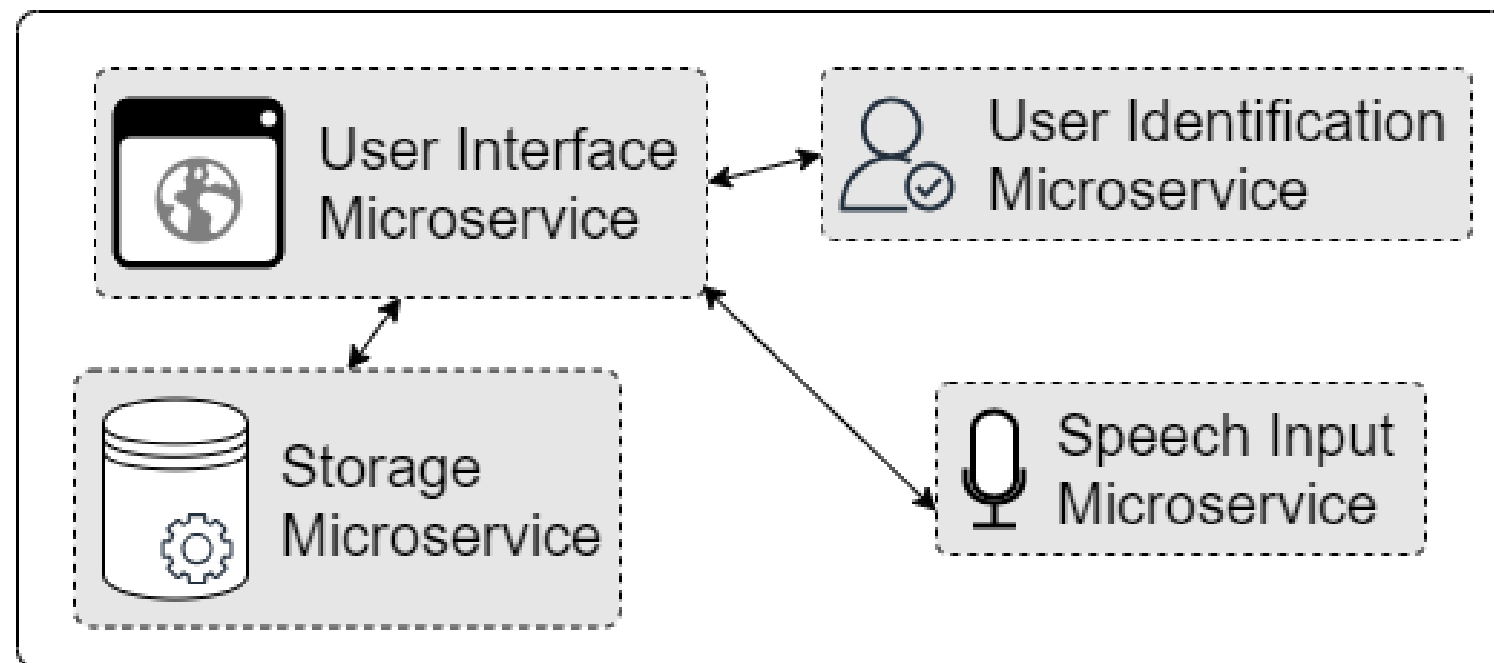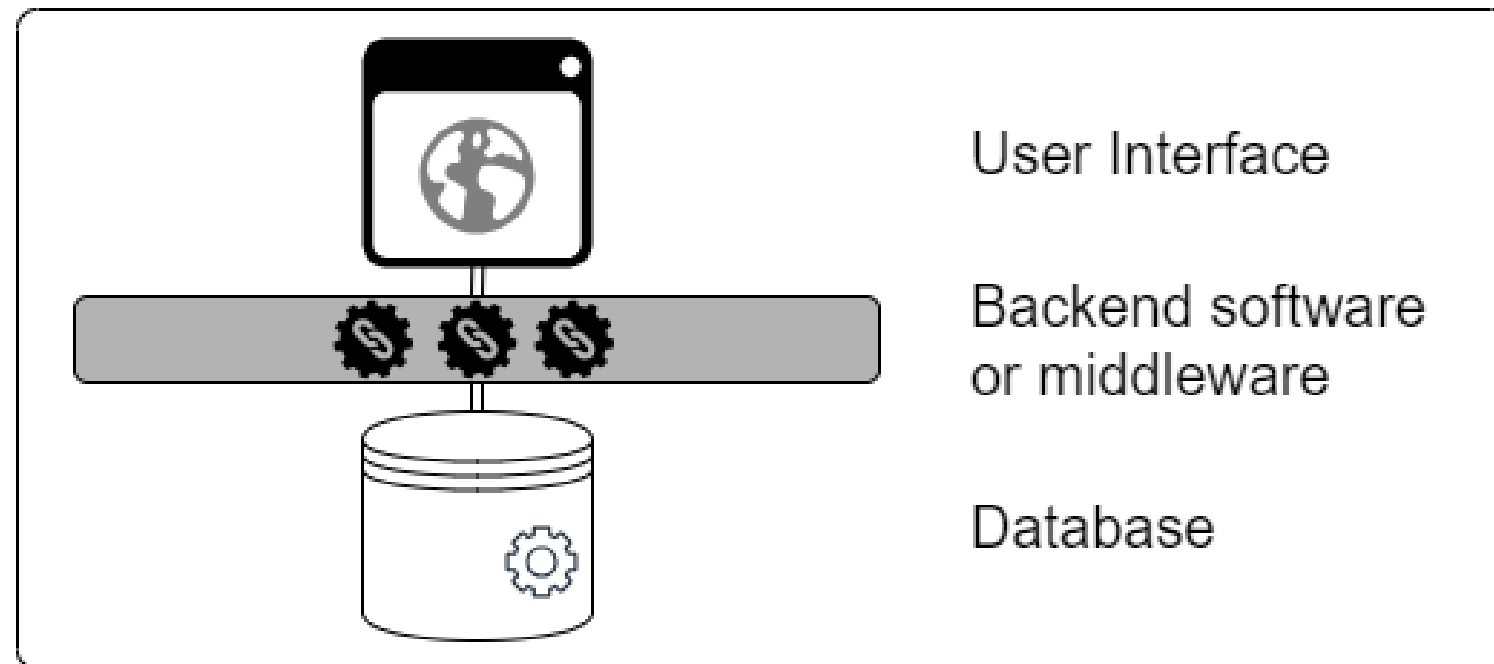Platform Architect and Freelance Instructor

About Me

- Experienced Platform Architect, projects in different sectors

- Into Kubernetes since 2018 with focus on scalable data platforms on-prem and using major cloud providers

- Focus on resilience and high availability

[1] https://de.linkedin.com/in/frank-heilmann-19556590

Modern Software Architecture

User Interface

Backend software
or middleware

Database

User Interface
Microservice

User Identification
Microservice

Storage
Microservice

Speech Input
Microservice

- Traditional architecture: **monoliths**

- Single applications, typically run independent from other applications
  - Hard to maintain and to update

- Modern architecture: constructed from independent building blocks: **microservices**
  - Can be independently maintained and updated

- Ideally suited for cloud computing

Containerization and Kubernetes
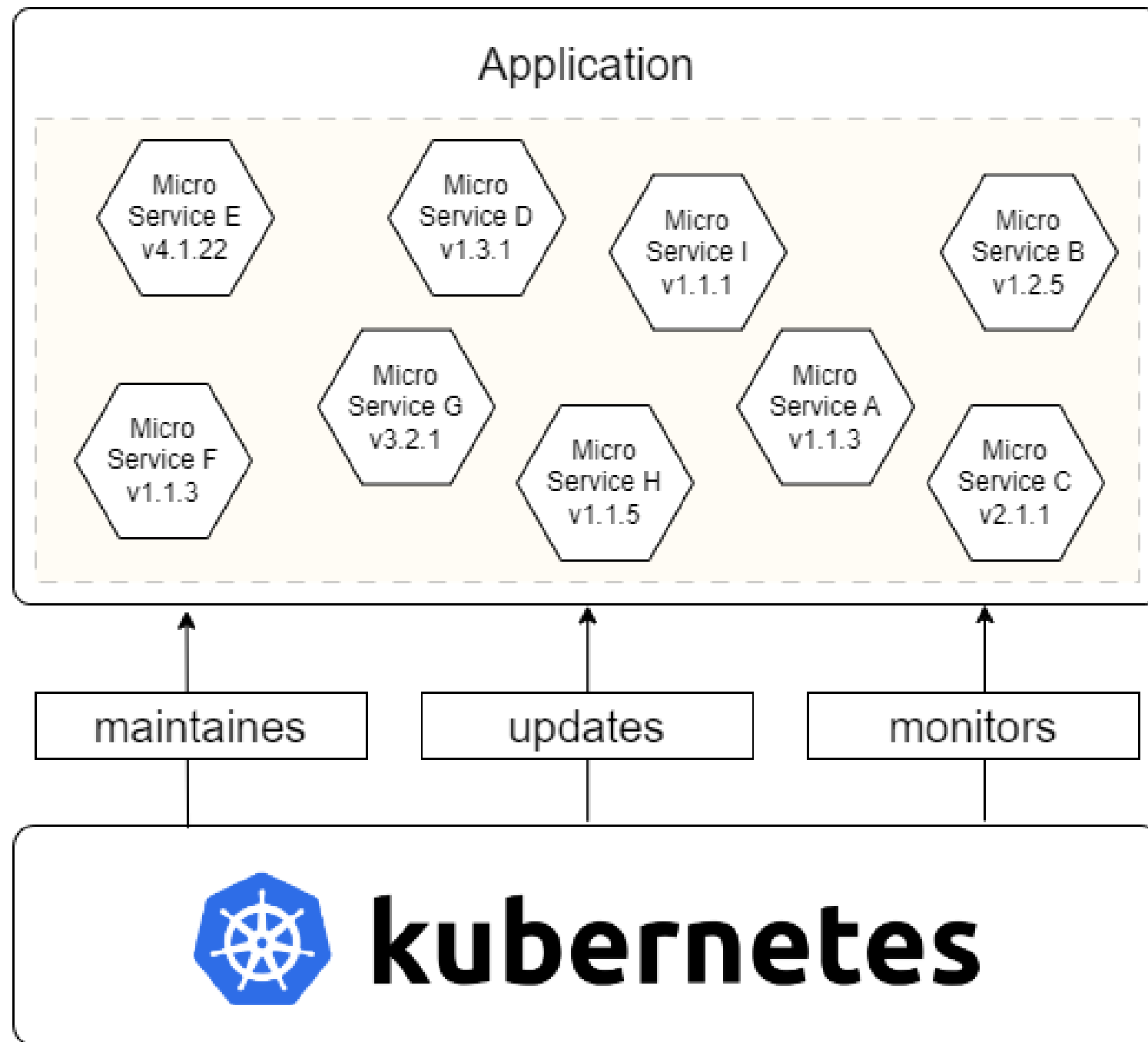
Application

- Modern applications consist of potentially thousands building blocks

- Deployed via **containers**!
  - Each building block is *delivered* in an individual container

- **Kubernetes** keeps track of all containers

Cloud Nativeness and Kubernetes

Cloud Provider

cloud-native application

virtual machines

storage
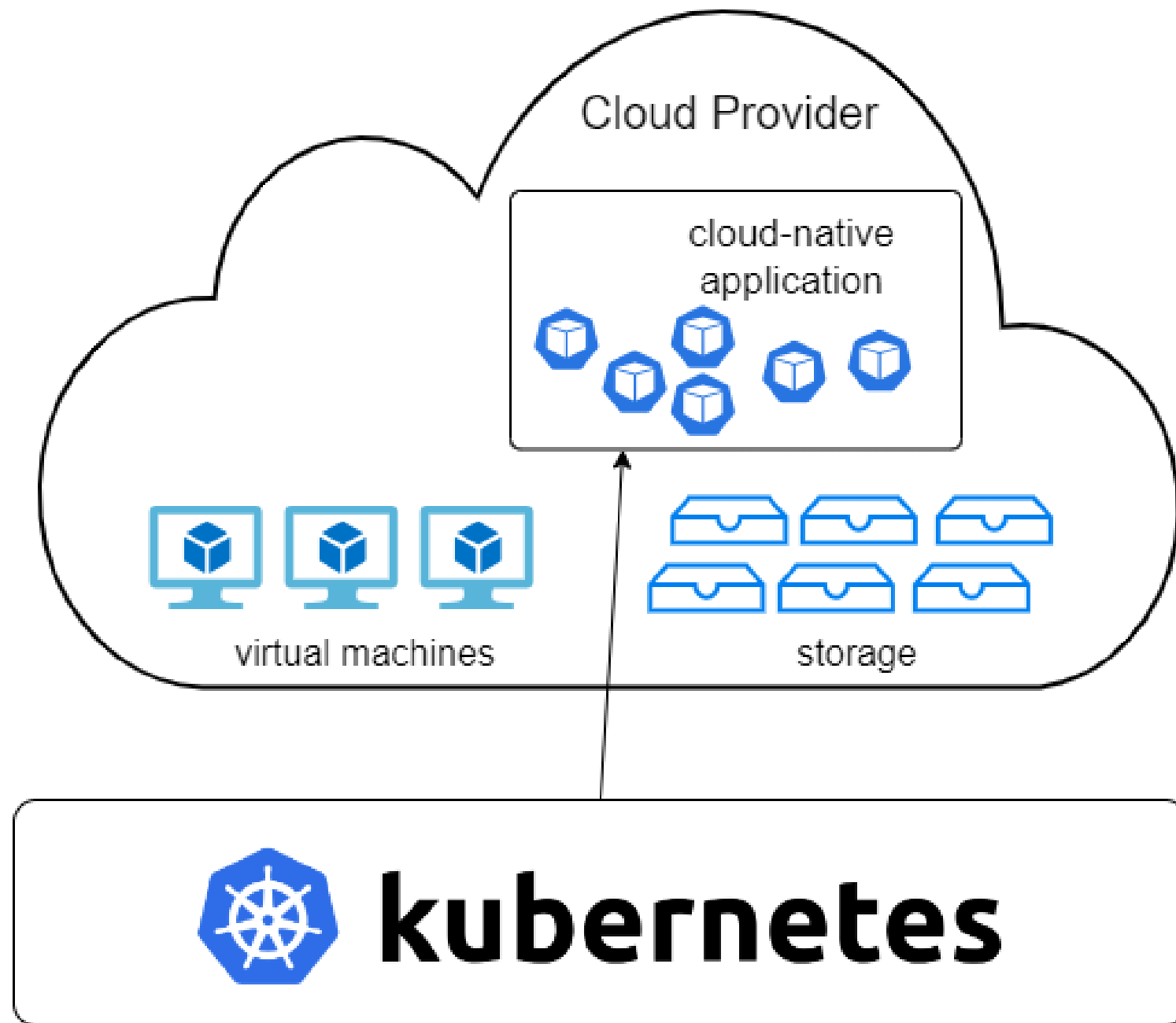
kubernetes

- *Cloud-native*: a way to build & deploy applications in the cloud

- **Cloud-native applications** are designed to be scalable

- **Kubernetes** is cloud-native:
  - simplifies deploying and managing containers
  - enables easy scaling of applications

Kubernetes Distributions and Cloud Offerings

- Kubernetes is an open-source project by Google

- Everyone can download, install, and use it

- We can also use Kubernetes with all cloud providers as a managed service

- "Kubernetes" --> "K *ubernete* s" --> "K8s"

Let's practice!

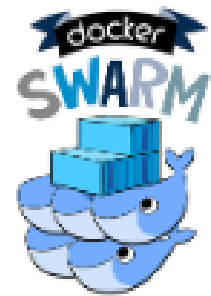INTRODUCTION TO KUBERNETES

Docker and Kubernetes

INTRODUCTION TO KUBERNETES

Frank Heilmann

Platform Architect and Freelance Instructor

Container Orchestration Tools

- Modern software stacks typically consist of potentially thousands of individual containers

- Managing all these containers is known as **Container Orchestration,** several **container orchestration tools** exist

- Kubernetes has an estimated market share well above 95%

Kubernetes for Orchestration

- Kubernetes solves the typical challenges of container orchestration, e.g.,
  - scheduling and networking (where to deploy a container and how to connect them)

  - how to attach storage to a container

- To do that, Kubernetes interacts with **Container Engines**.
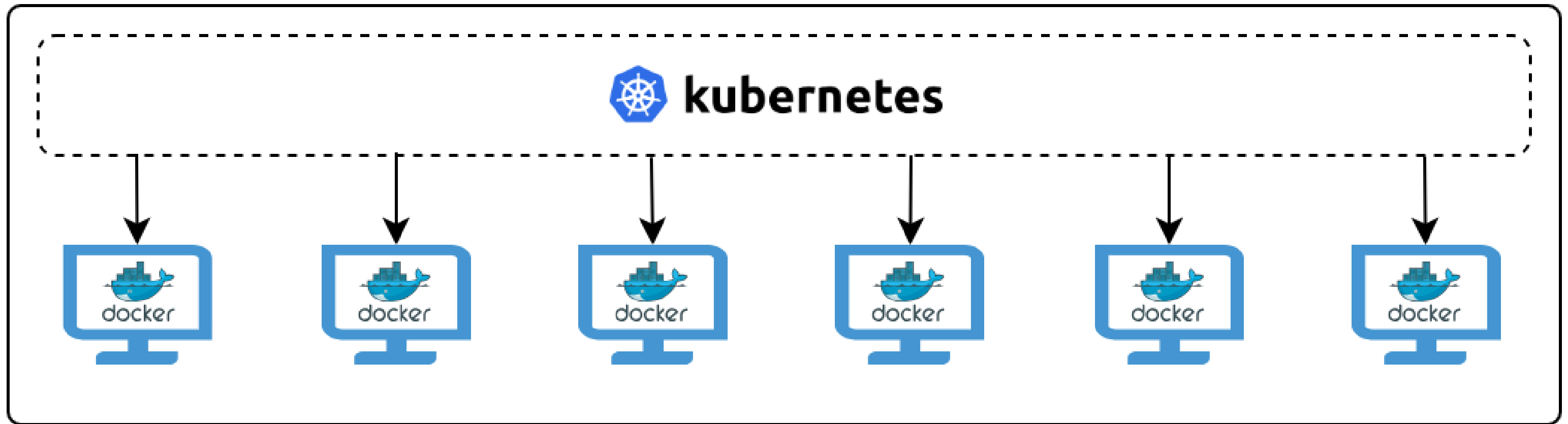
# The Relationship between Docker and Kubernetes

- Often, **Docker** is your container engine of choice

- Kubernetes interacts with Docker as a container engine to schedule and maintain containers

- Docker is typically used for two tasks:
  - creating and updating Docker images
  - starting containers from such images

- Kubernetes never creates Docker images, you use Docker for this

Kubernetes Manifests

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 5
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.25.4
        ports:
        - containerPort: 80
```

- Kubernetes objects (e.g., containers) are described in so-called **Kubernetes Manifests**

- Manifests are **YAML files** that describe which objects you want, how they should be configured, where they should be scheduled, and a lot more

- Manifest are **declarative**, i.e., you describe **what** you want, or which **state** to achieve

- They are **not imperative**, you do **not describe how to achieve it**

kubectl
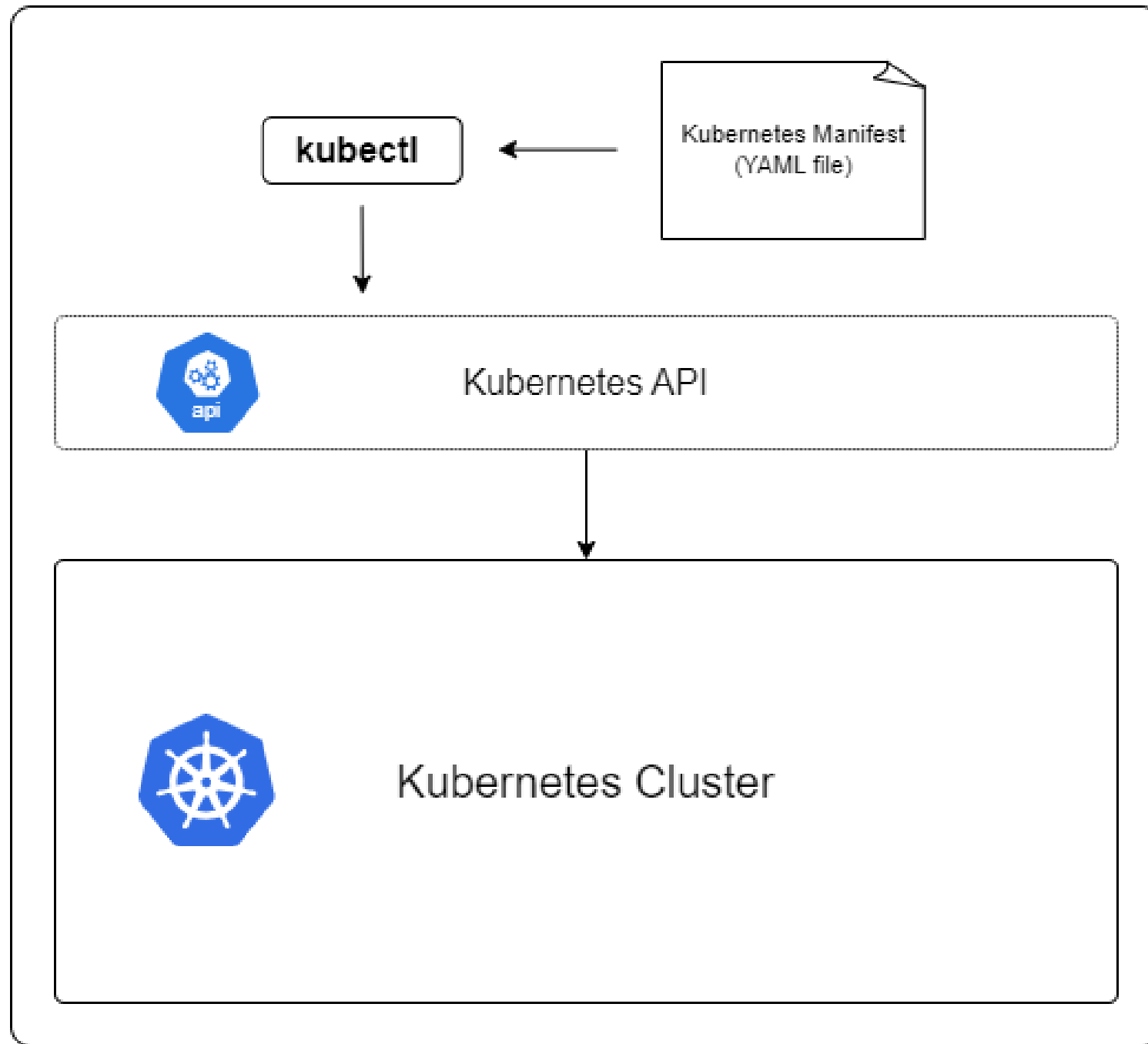
- `kubectl` is a command line tool to interact with Kubernetes

- `kubectl` comes with many commands and options

- `kubectl` reads your Manifest, sends them to Kubernetes via its API, and Kubernetes will compute what to do to achieve the state you want

- Pronounced *cube cuddle* ;-)

Let's practice!

INTRODUCTION TO KUBERNETES

Kubernetes Architecture

## INTRODUCTION TO KUBERNETES

Frank Heilmann

Platform Architect and Freelance Instructor

Kubernetes Overview

- Kubernetes is built from many elements:

- Most important ones, from larger to smaller:
  - **Clusters** and **Control Planes**

  - **Nodes**

  - **Pods**

- Network connectivity through **Services**

Kubernetes Cluster and the Control Plane

Kubernetes Cluster

Kubernetes Control Plane

node

node

node

...
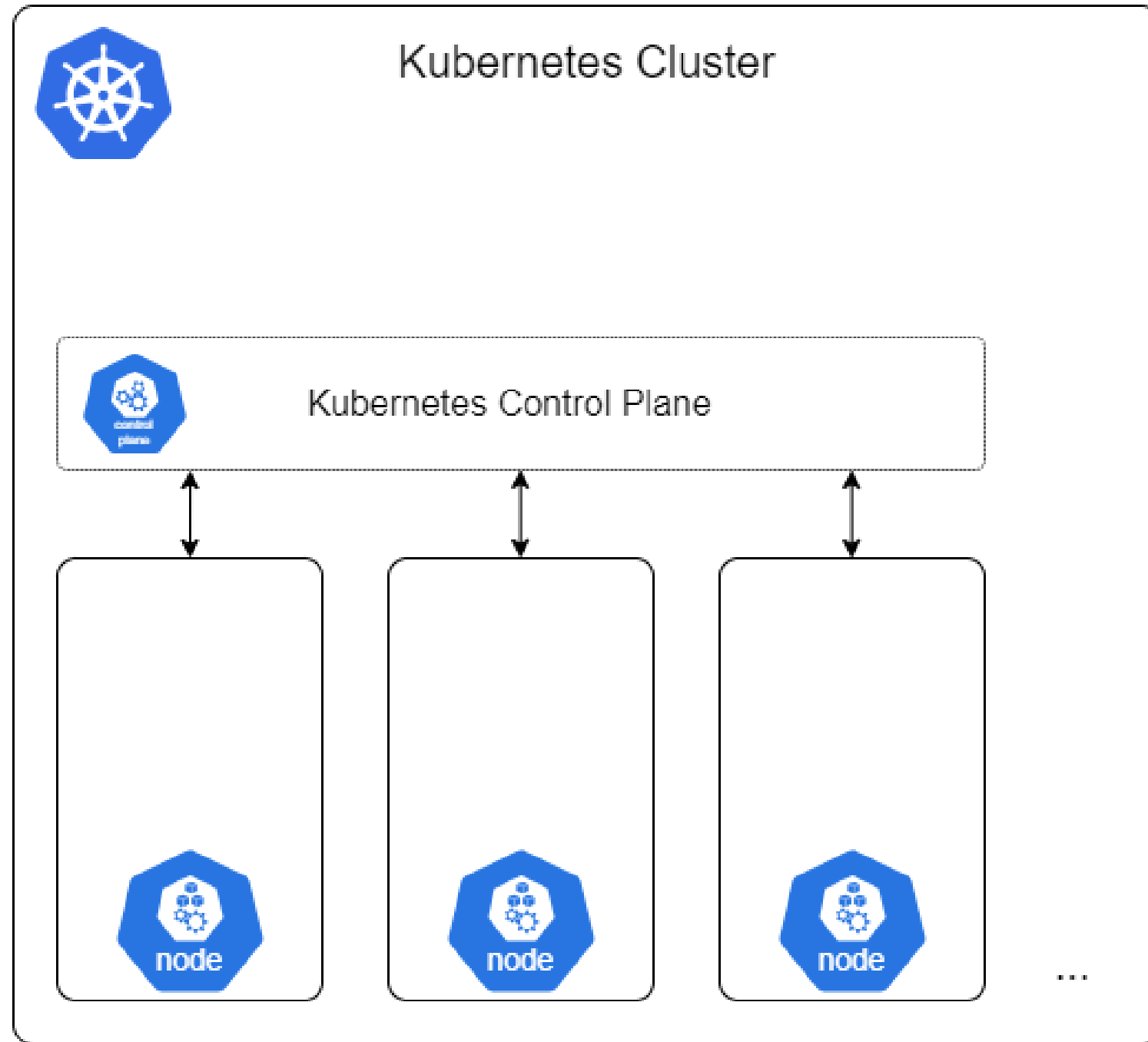
- A Kubernetes Cluster is a set of connected computers (or **Nodes**)

- Servers in a datacenter, virtual machines in the cloud

- The Kubernetes **Control Plane** manages these nodes
  - consists of many components, that can run on any node in the cluster

Kubernetes Nodes
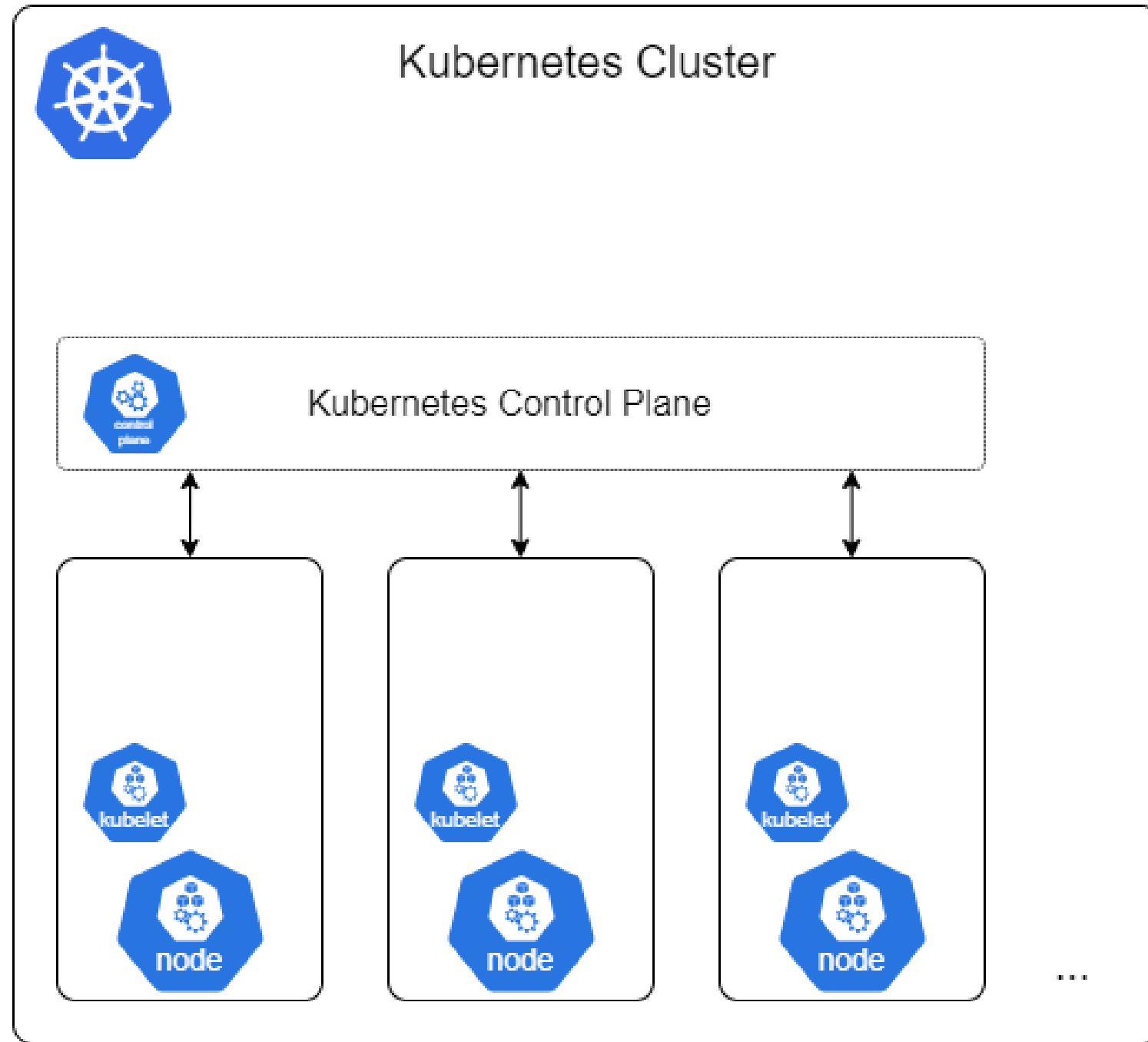
- A Kubernetes Node typically runs Linux + container engine (Docker)

- Nodes are also called **worker machines**

- Nodes run **Kubernetes Kubelet**
    - ensures containers run in so-called *Pods*

Kubernetes Pods
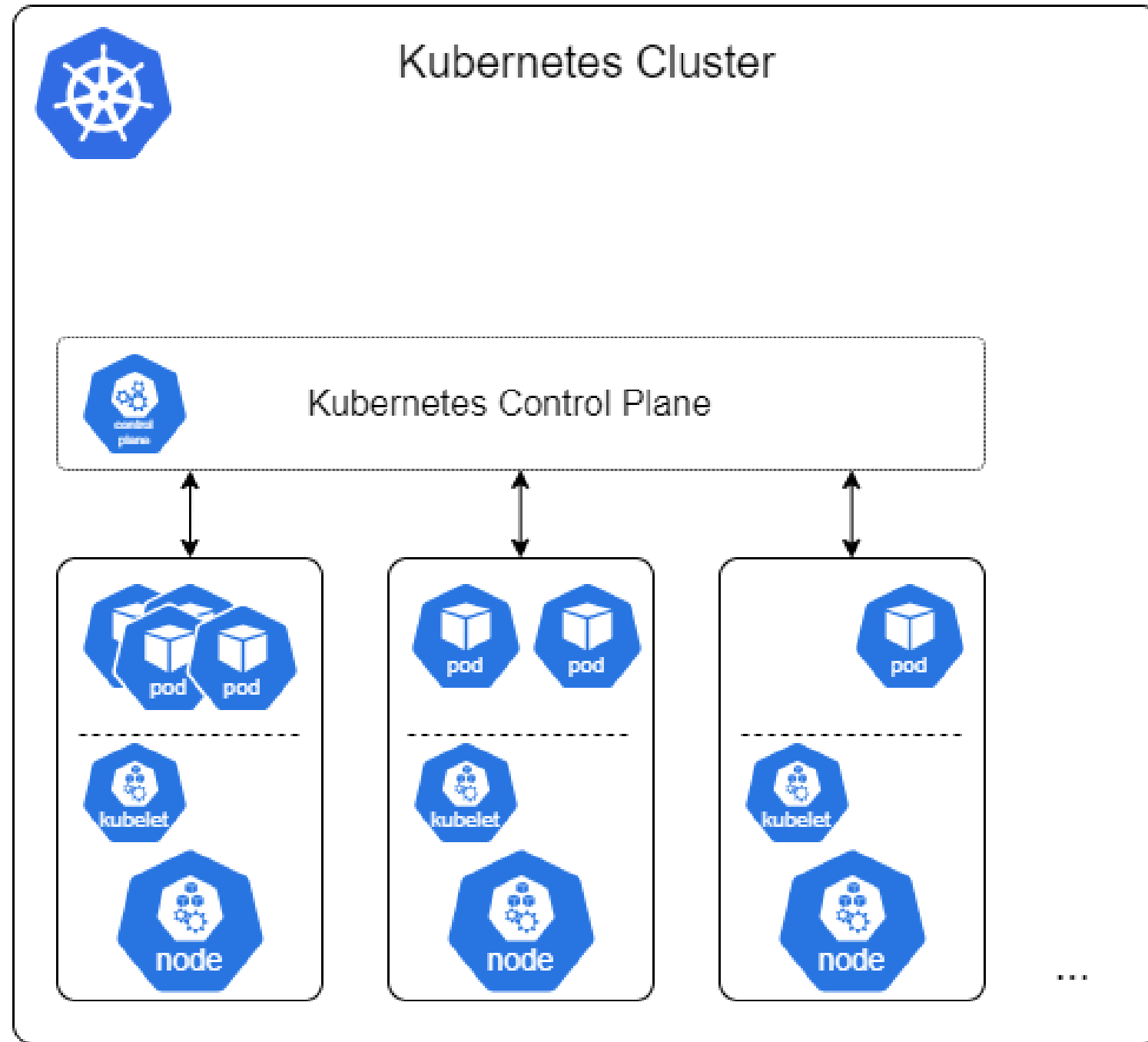
- Kubernetes **Pod**
  - Smallest unit that you can deploy
  - A Pod is a set of one or more containers
- The containers in a Pod belong together logically, share storage and network
- Pods are *ephemeral*:
  - Pods can be stopped and recreated and any point in time.
  - Pods can moved to other nodes at any point in time.

Kubernetes Services
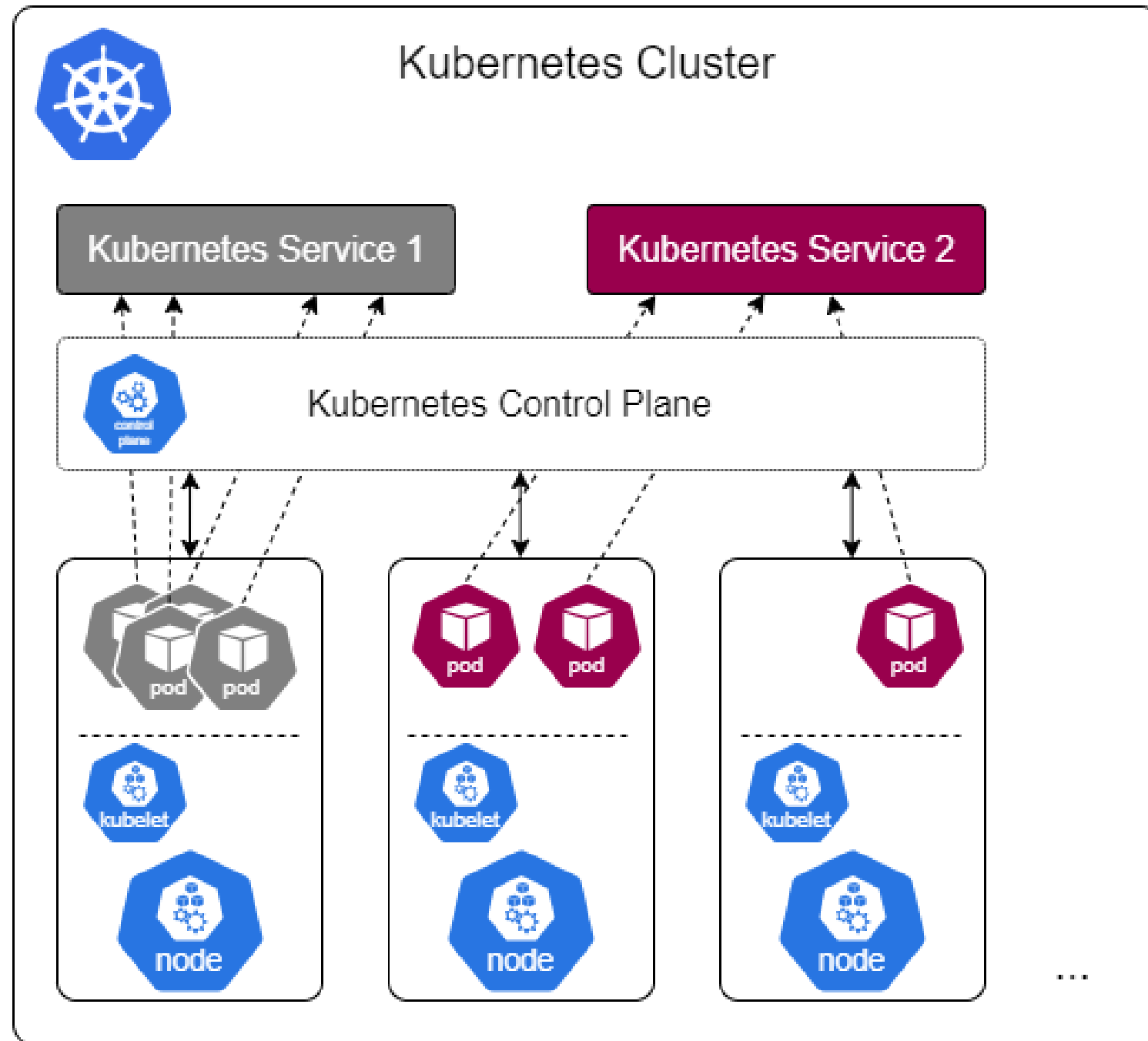
- **Kubernetes Service**: resource for exposing network connectivity

- Required to connect to Pod from outside, or to communicate between Pods

- Reason: Pods may get re-deployed any time, and will
    - Receive a new IP address

- Services are not *ephemeral*, they offer stable network connectivity

Kubernetes Cheat Sheet

- Kubernetes **Cluster**: set of connected computers (Nodes) configured to run Kubernetes

- Kubernetes **Control Plane**: manages the Nodes in a Cluster

- Kubernetes **Nodes**: also called "worker machines", running Linux and a container engine

- Kubernetes **Pods**: a set of one or more containers, the smallest deployable unit

- Kubernetes **Services**: a resource for exposing network connectivity, required to connect to Pods from outside, and for communication between Pods

Let's practice!

INTRODUCTION TO KUBERNETES