Microsoft

WorkshopPLUS - Containers as Infrastructure

Microservices and Containers Orchestration Platforms

Microsoft Services

Objectives

- Learn the role of a container orchestration and understand options of orchestrators.
- Docker Swarm, Mesos, Kubernetes, and Azure Service Fabric and understanding their capabilities and limitations.

Microsoft

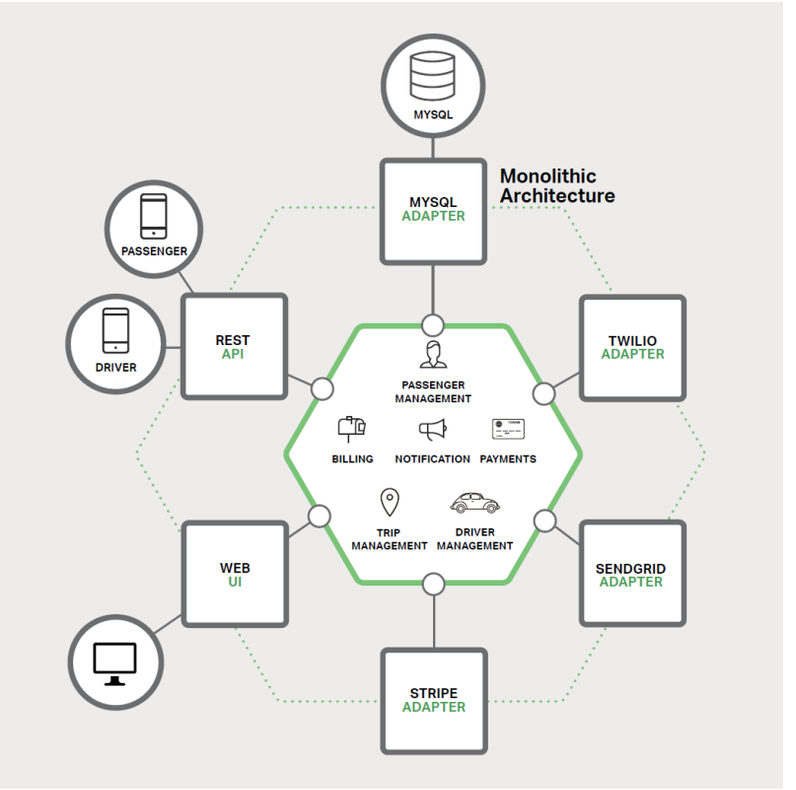
Microservices and Containers Orchestration Platform

Introduction to Microservices

Microsoft Services

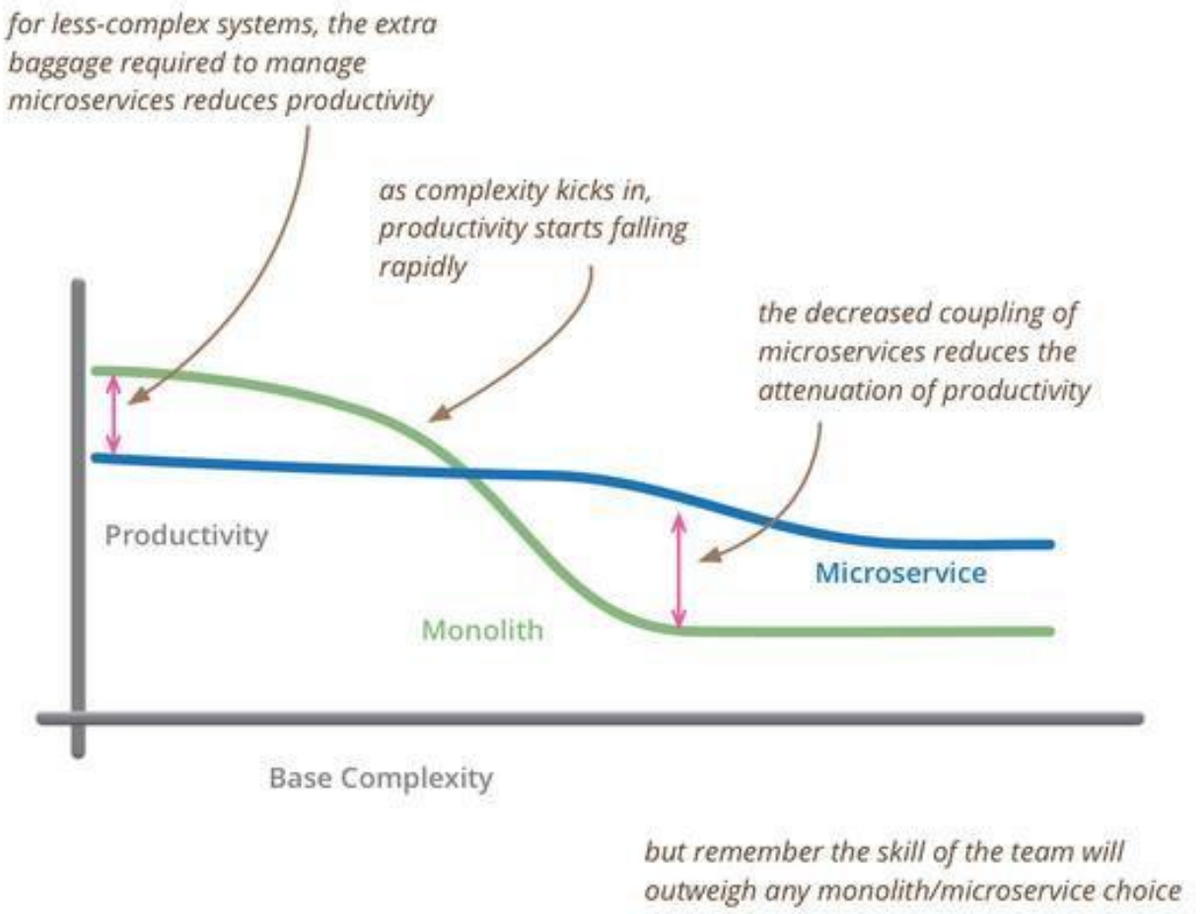
What is a Monolith?

- Most applications built in the last 15 years
- A large application core, composed of a single key technology
- Typically surrounded by a layer of adapters that expose its functionality to the outside world and provide access to dependencies



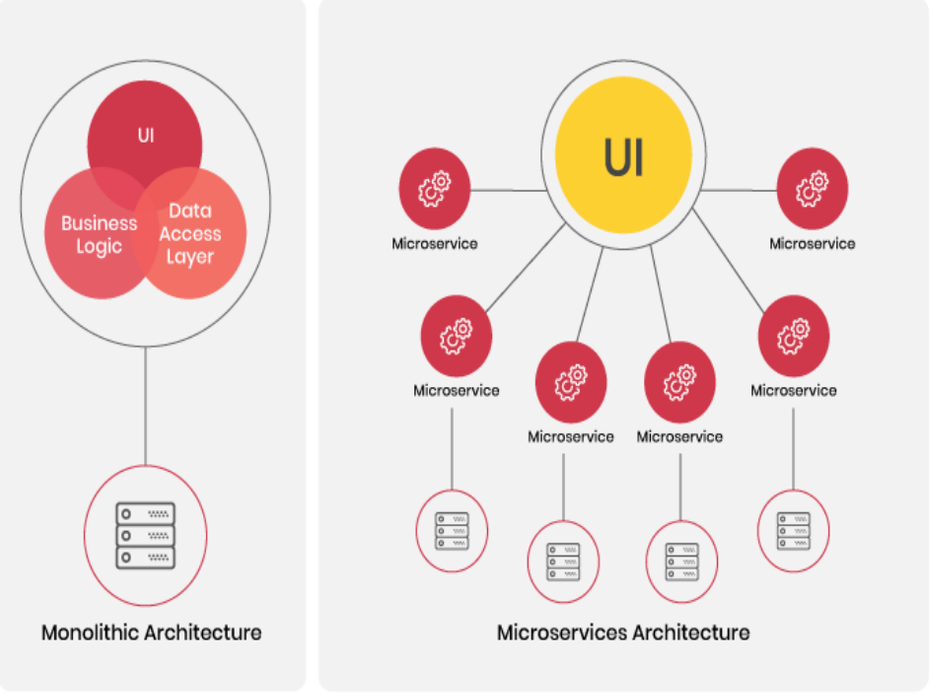
Shortcomings of the Monolith

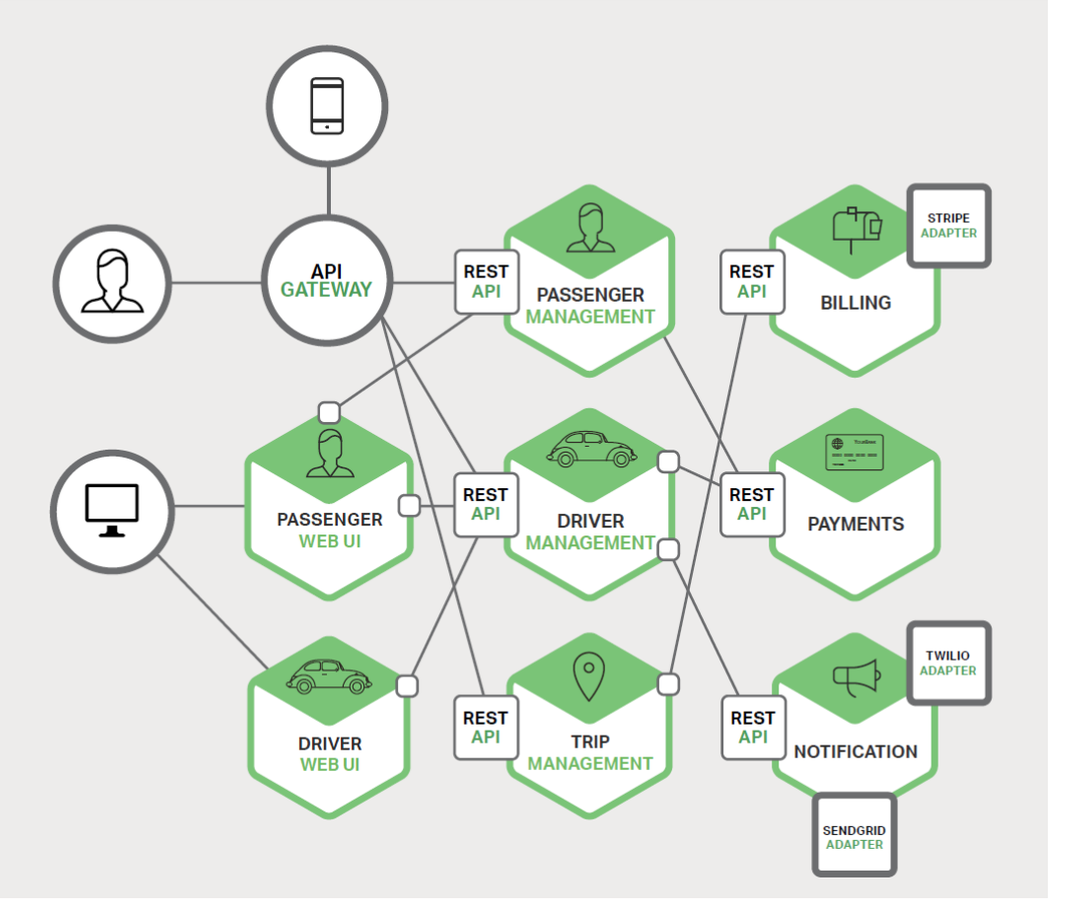
Over time, application complexity grows, causing applications to grow stale and clunky.



What are Microservices?

- An approach to application development
- Large application built around a collection of services
- Distributed architecture

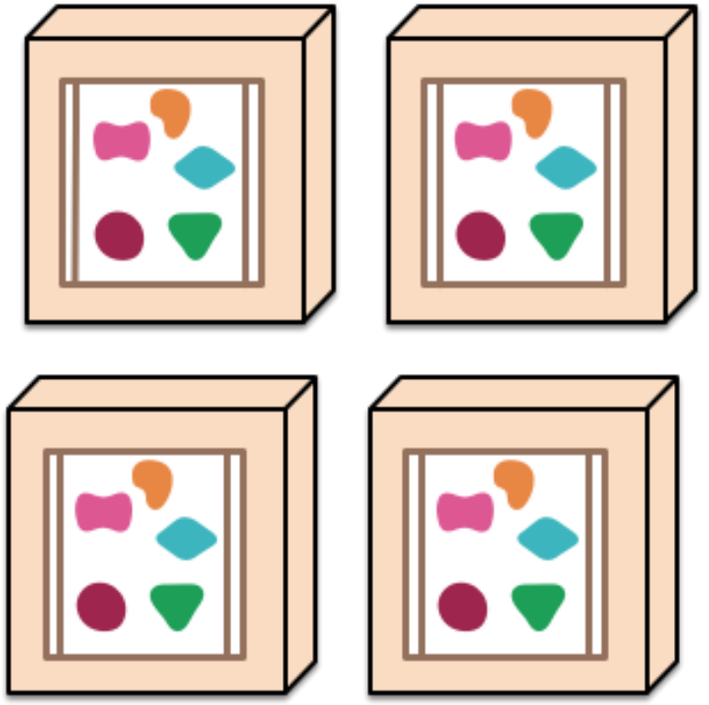




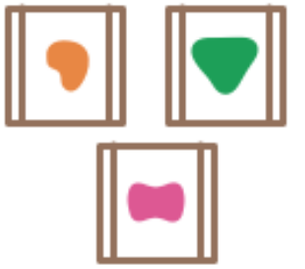
Microservices Architecture

A monolithic application puts all its functionality into a single process...

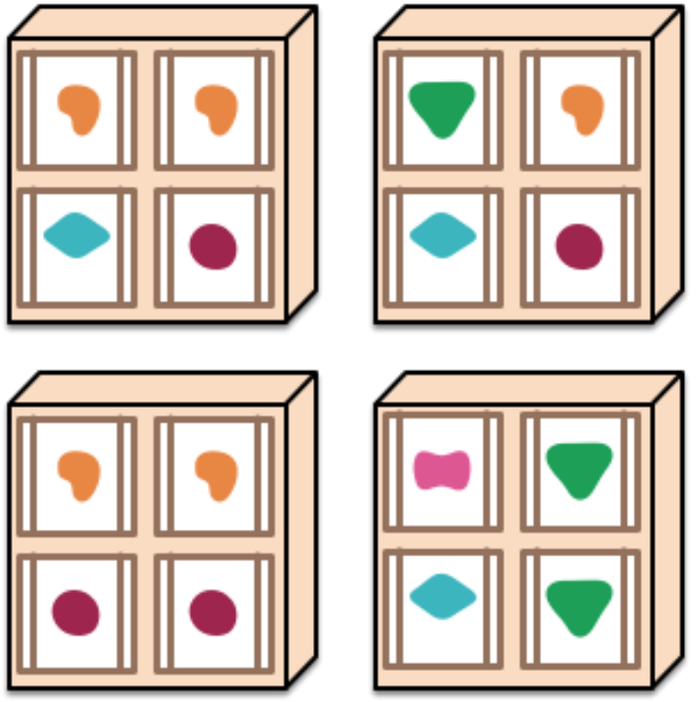
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...

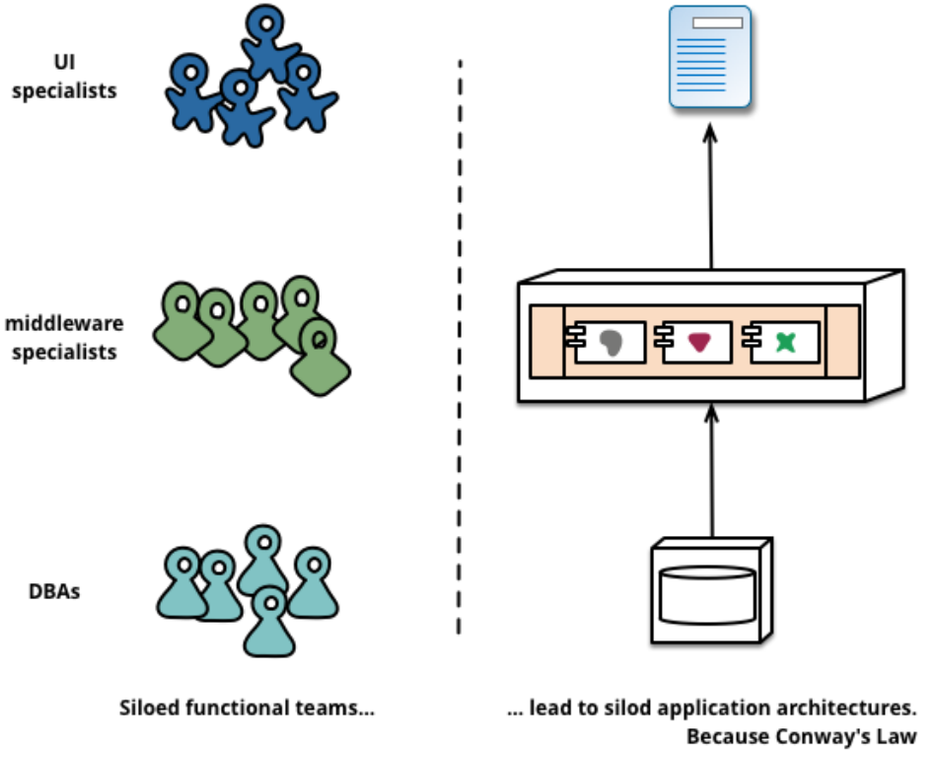


... and scales by distributing these services across servers, replicating as needed.

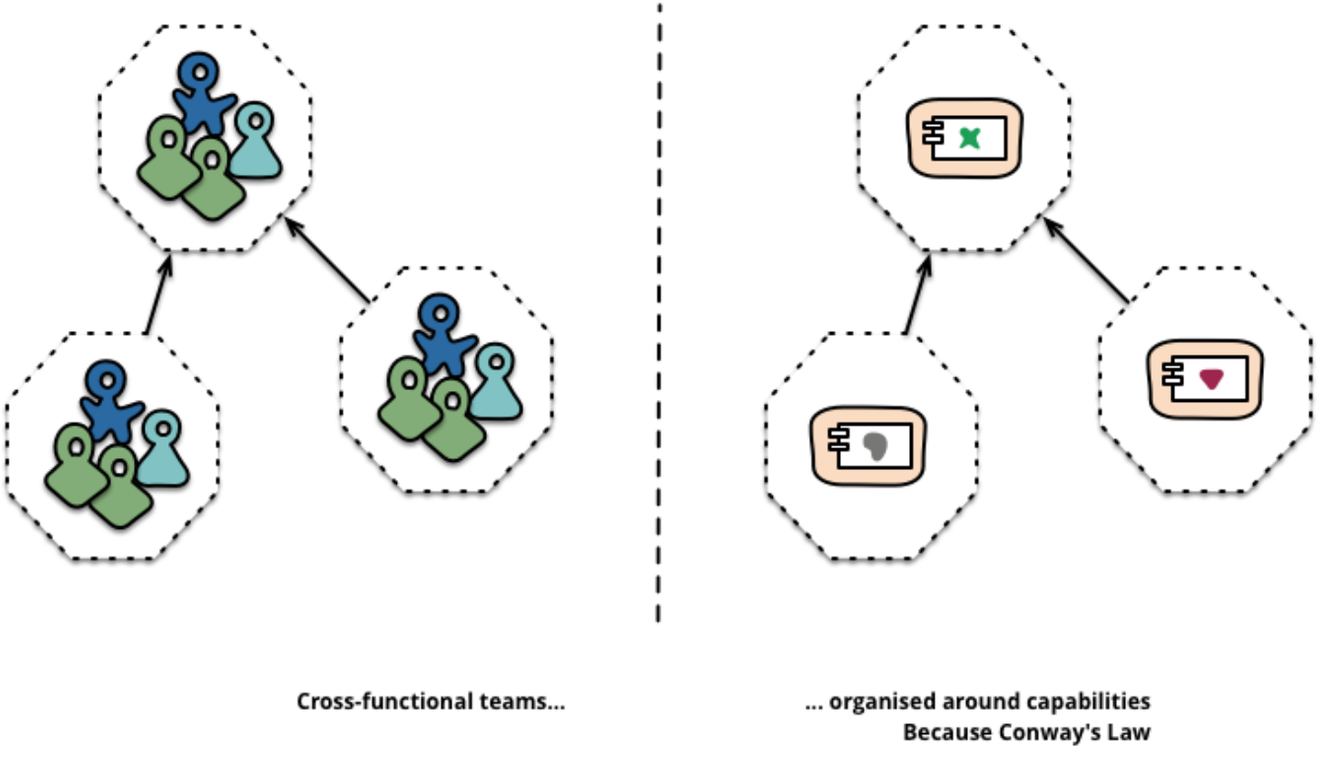


Microservices Architecture (Cont.)

"Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure." -- Melvyn Conway, 1967



Monolithic



Microservices

Microservices and Containers

Microservices

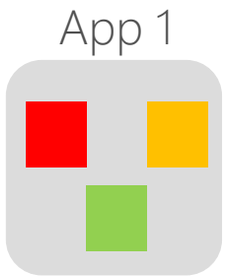
- A large application core, composed of a single key technology
- Typically surrounded by a layer of adapters that expose its functionality to the outside world and provide access to dependencies
- Commonly deployed in containers

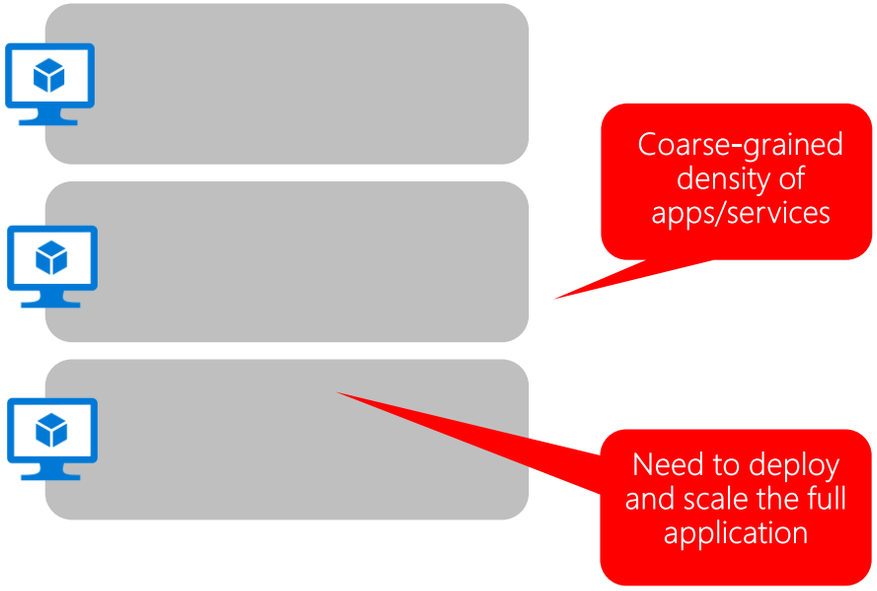
Containers

- Provide fine-grained execution environment
- Provide a high degree of isolation
- Provide fast initialization and execution
- Leverage orchestrators to manage microservice-based apps
- Using container tools, help simplify complexities inherent in applications

Traditional application approach

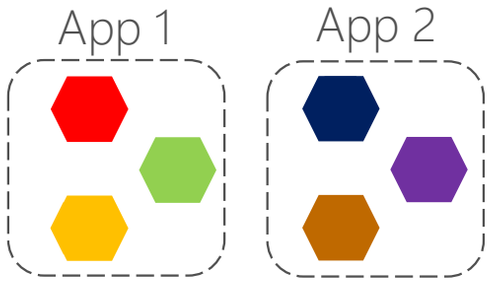
- A traditional application has most functionality within a few componentized processes with layers and libraries.
- Scales by cloning the app on multiple servers/VMs

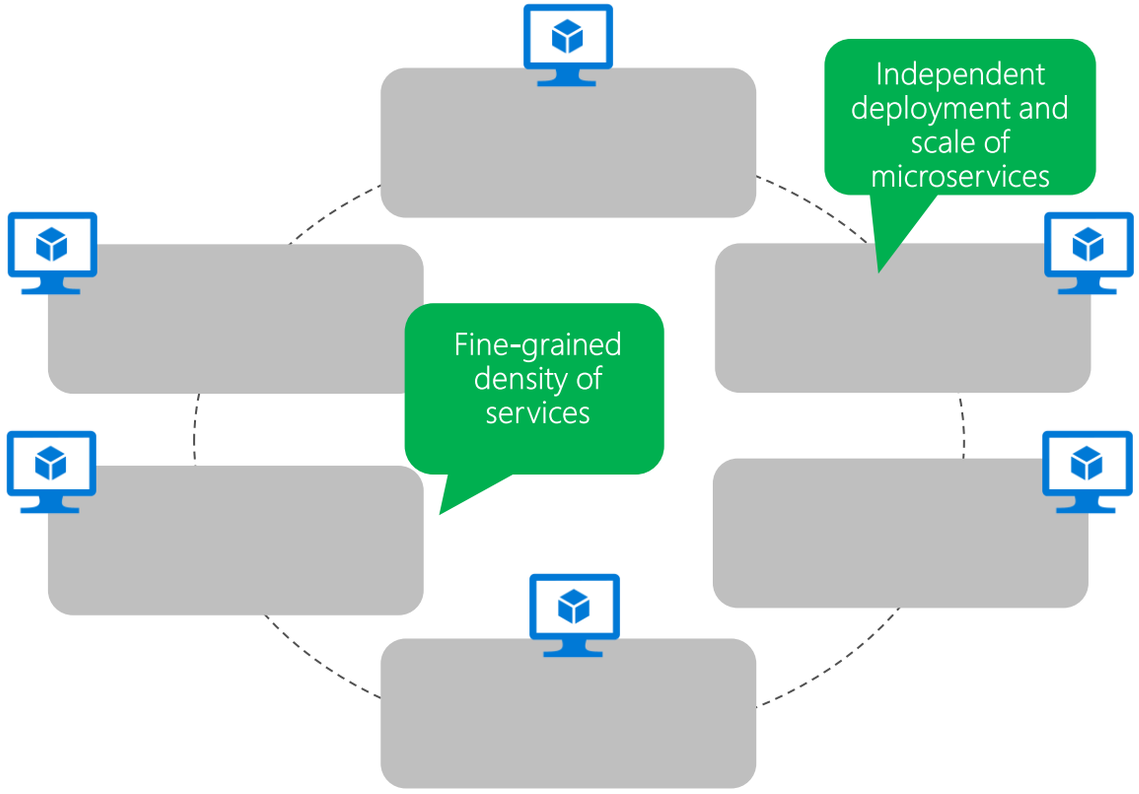




Microservices application approach

- A microservice application segregates functionality into separate smaller services.
- Scales out by **deploying each service independently** with multiple instances across servers/VMs





Microsoft

Microservices and Containers Orchestration Platform

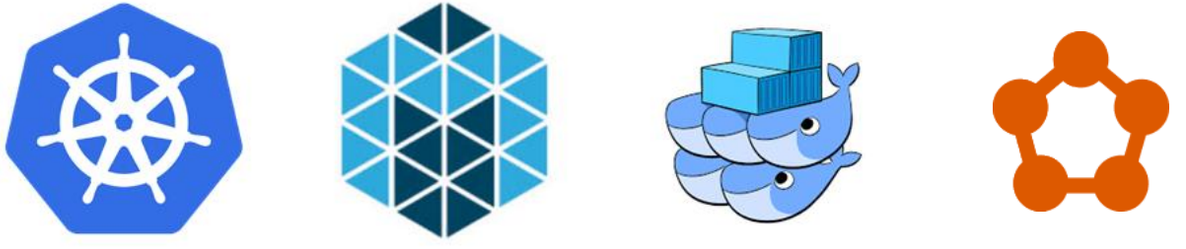
Containers Orchestration Core Concepts

Microsoft Services

Why Container Orchestration?

Container orchestration is required to transition from deploying containers individually on a single host, to deploying complex multi-container apps on many machines. It requires a distributed platform, independent from infrastructure, that stays online through the entire lifetime of your application, surviving hardware failure and software updates.

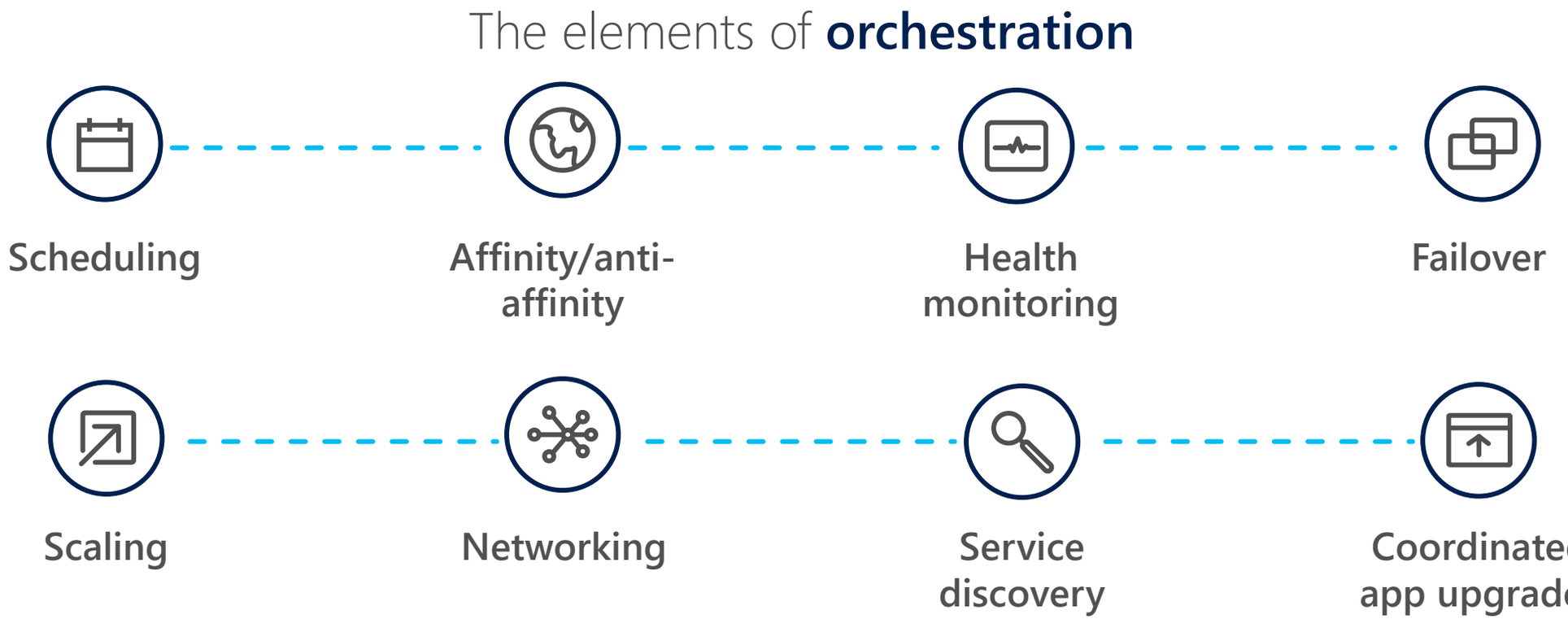
Some of the main Container Orchestration Platforms:



- Kubernetes
- Mesos
- Docker Swarm
- Azure Service Fabric

What is Container Orchestration?

Container orchestration is all about managing the lifecycles of containers, especially in large, dynamic environments. Container orchestration engines manage how multiple containers are created, upgraded, and made highly available.



Microsoft

Microservices and Containers Orchestration Platform

Introduction to Docker Swarm

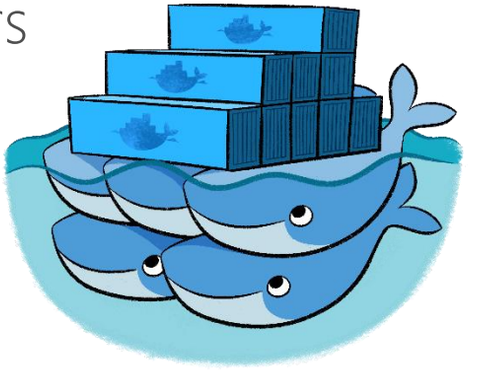
Microsoft Services

Docker Swarm

Docker swarm is Docker's own containers orchestration system. Docker Swarm provides an environment for deploying containerized workloads across a pooled set of Docker hosts. A Swarm consists of multiple Docker hosts which run in swarm mode and act as managers and workers. Manager nodes perform orchestration and cluster management. Worker nodes receive and execute tasks from the manager nodes.

Features of Docker Swarm:

- Coordinate between containers and allocate tasks to groups of containers
- Perform health checks and manage lifecycle of individual containers
- Provide redundancy and failover in case nodes experience failure
- Scale the number of containers up and down depending on load
- Perform rolling updates of software across multiple containers



Docker Swarm Mode

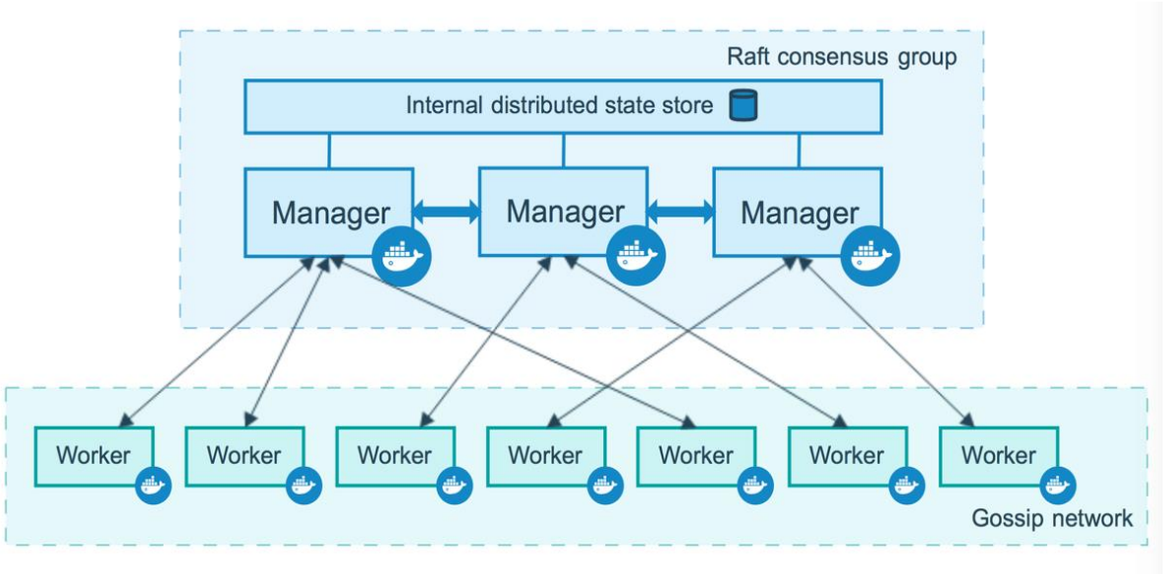
Swarm mode is a Docker feature that provides built in container orchestration capabilities, including native clustering of Docker hosts and scheduling of container workloads.

Node: A node is an instance of a Swarm. Nodes can be distributed on-premises or in public clouds.

Swarm: a cluster of nodes (or Docker Engines). In Swarm mode, you orchestrate services, instead of running container commands

Manager Nodes: These nodes receive service definitions from the user, and dispatch work to worker nodes. Manager nodes can also perform the duties of worker nodes.

Worker Nodes: These nodes collect and run tasks from manager nodes. Service: A service specifies the container image and the number of replicas.



Swarm Mode Nodes: Managers and Workers

Docker Swarm Mode

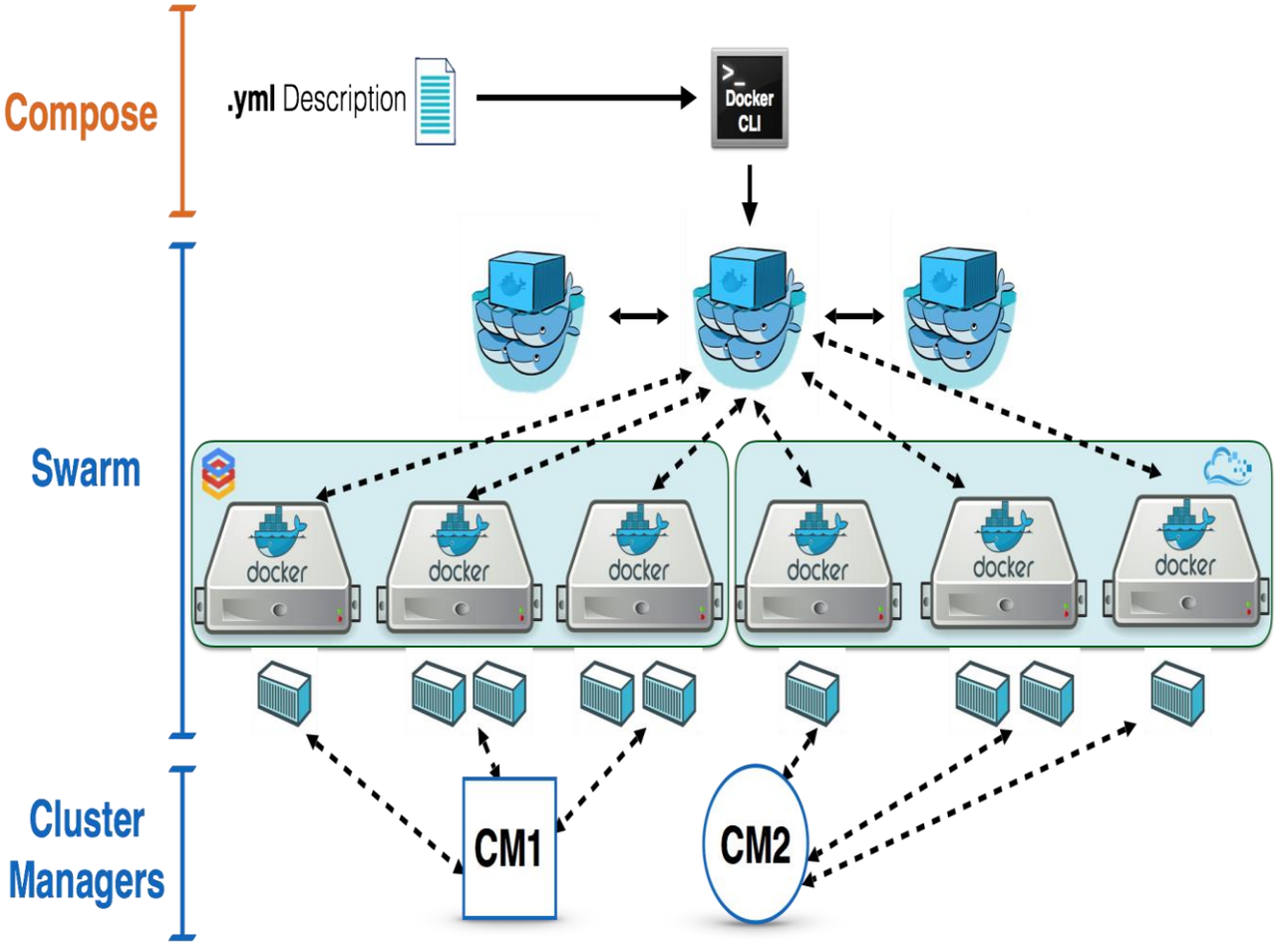
Swarm mode is a Docker feature that provides built in container orchestration capabilities, including native clustering of Docker hosts and scheduling of container workloads.

Node: A node is an instance of a Swarm. Nodes can be distributed on-premises or in public clouds.

Swarm: a cluster of nodes (or Docker Engines). In Swarm mode, you orchestrate services, instead of running container commands

Manager Nodes: These nodes receive service definitions from the user, and dispatch work to worker nodes. Manager nodes can also perform the duties of worker nodes.

Worker Nodes: These nodes collect and run tasks from manager nodes. Service: A service specifies the container image and the number of replicas.



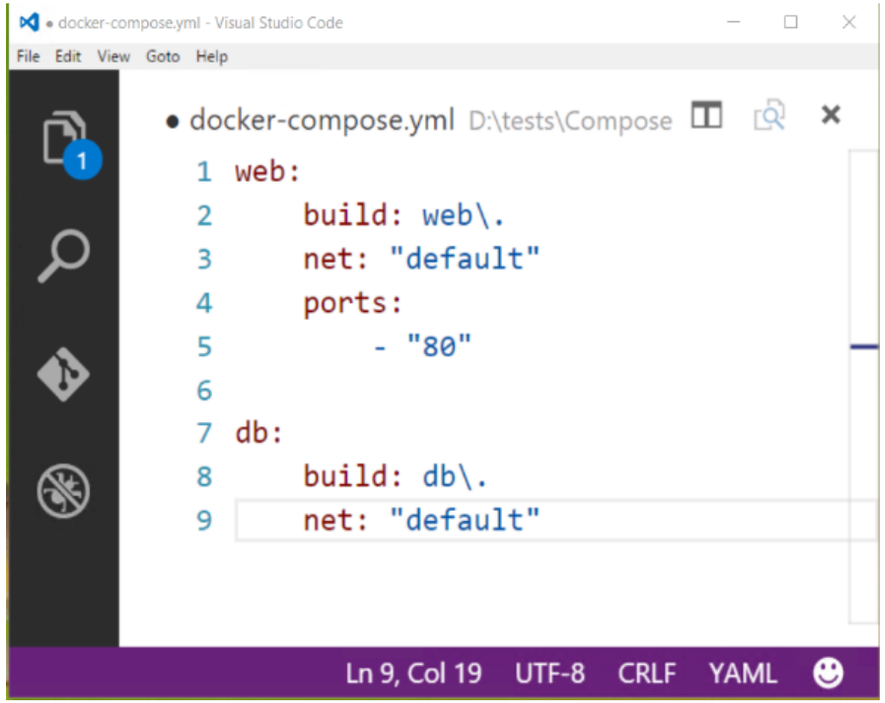
More Docker

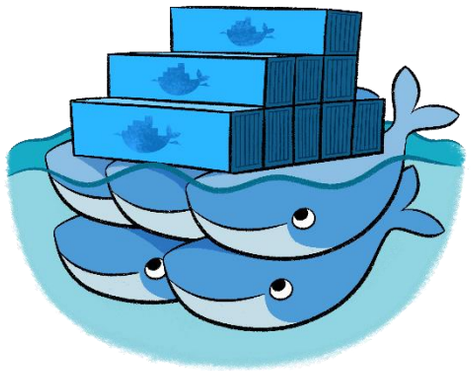
Docker Compose

- Define application as separate containers
- Manage different containers as a unit
- Scale parts of application as needed

Docker Swarm

- Aggregate container hosts
- Supports tagging, affinity/anti-affinity





Microsoft

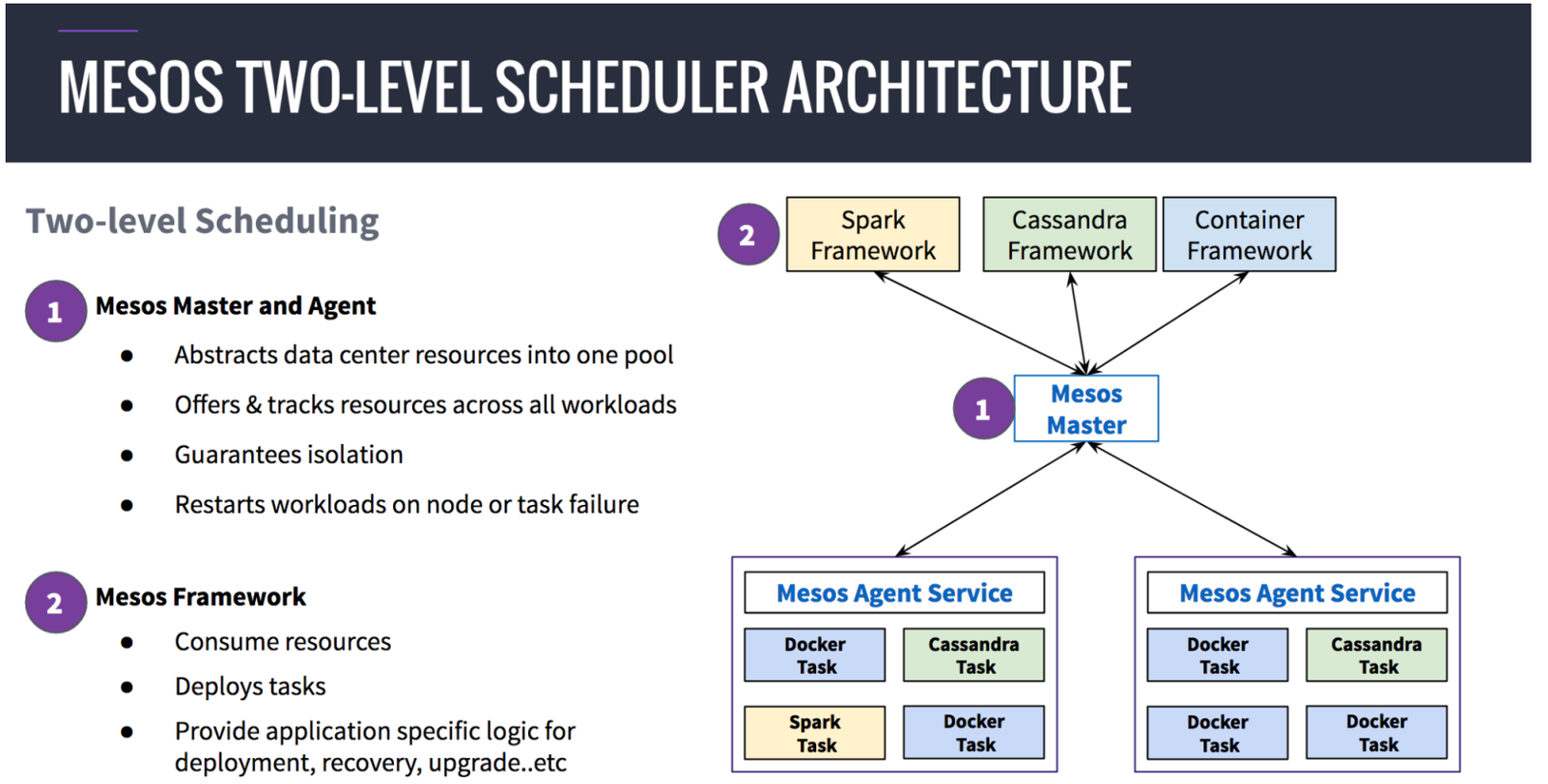
Microservices and Containers Orchestration Platform

Introduction to Mesos

Microsoft Services

Apache Mesos

- Apache Mesos is a cluster manager that provides efficient resource isolation and sharing across distributed applications, or frameworks. It can run Hadoop, Jenkins, Spark, Kubernetes and other frameworks on a dynamically shared pool of nodes.



Apache Mesos

Mesos was architected to solve for a very different set of challenges:

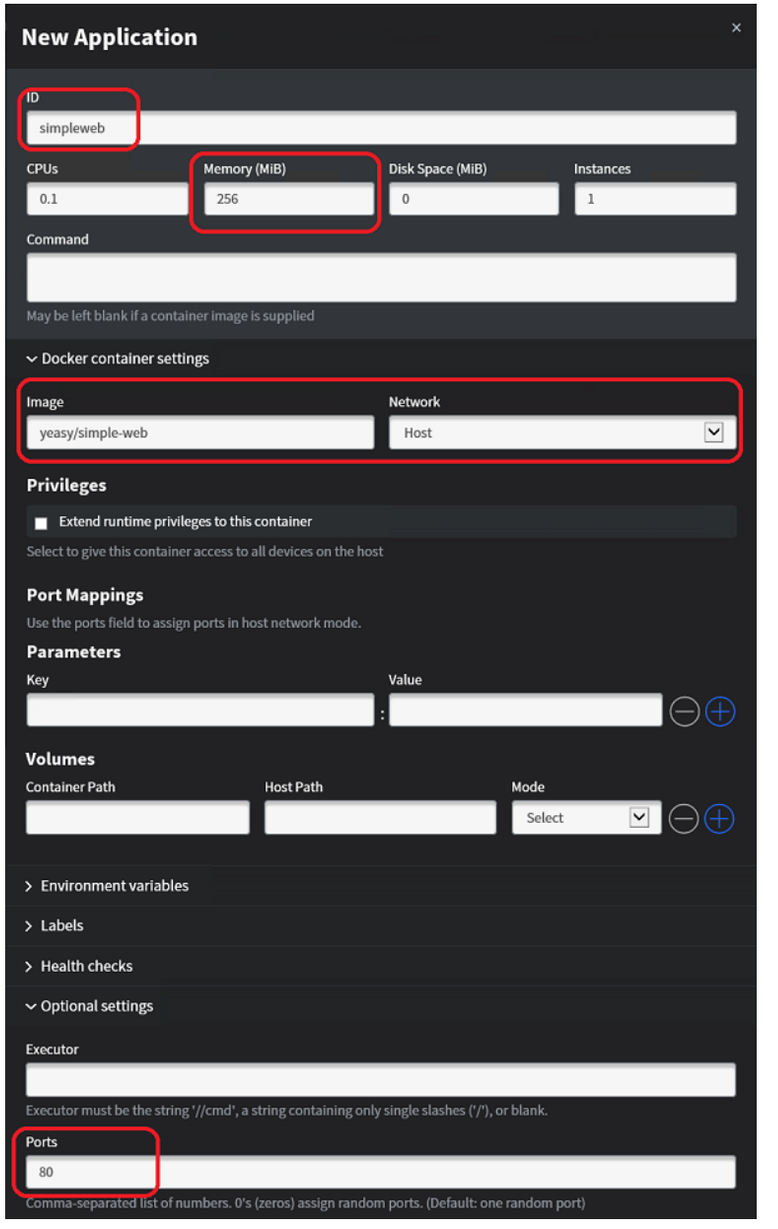
- **Abstract data center resources** into a single pool to simplify resource allocation while providing a consistent application and operational experience across private or public clouds
- **Co-locate diverse workloads** on the same infrastructure; such as analytics, stateless microservices, distributed data services and traditional apps to improve utilization and reduce cost and footprint
- **Automate day-two operations** for application-specific tasks such as deployment, self healing, scaling, and upgrades; providing a highly available fault tolerant infrastructure
- **Provide evergreen extensibility** to run new application and technologies without modifying the cluster manager or any of the existing applications built on top of it
- **Elastically scale** the application and the underlying infrastructure from a handful to tens of thousands of nodes

Mesos + Marathon

- Container orchestration is one example of a workload that can run on Mesos' modular architecture, and it's done using a specialized orchestration "framework" built on top of Mesos called Marathon. Marathon was originally developed to orchestrate app archives (like JARs, tarballs, ZIP files) in cgroup containers, and was one of the first container orchestrators to support Docker containers in 2014.
- When people compare Docker and Kubernetes to Mesos, they are comparing Kubernetes and Docker Swarm to Marathon running on Mesos.
- Apache Mesos can function as a container orchestration system, but its abilities allow it to extend beyond containers to microservices, Big Data, analytics engines, and beyond.

Mesos/Mesosphere + Marathon

- Aggregates container hosts
- Web based UI
- Service Launch and Discovery



Microsoft

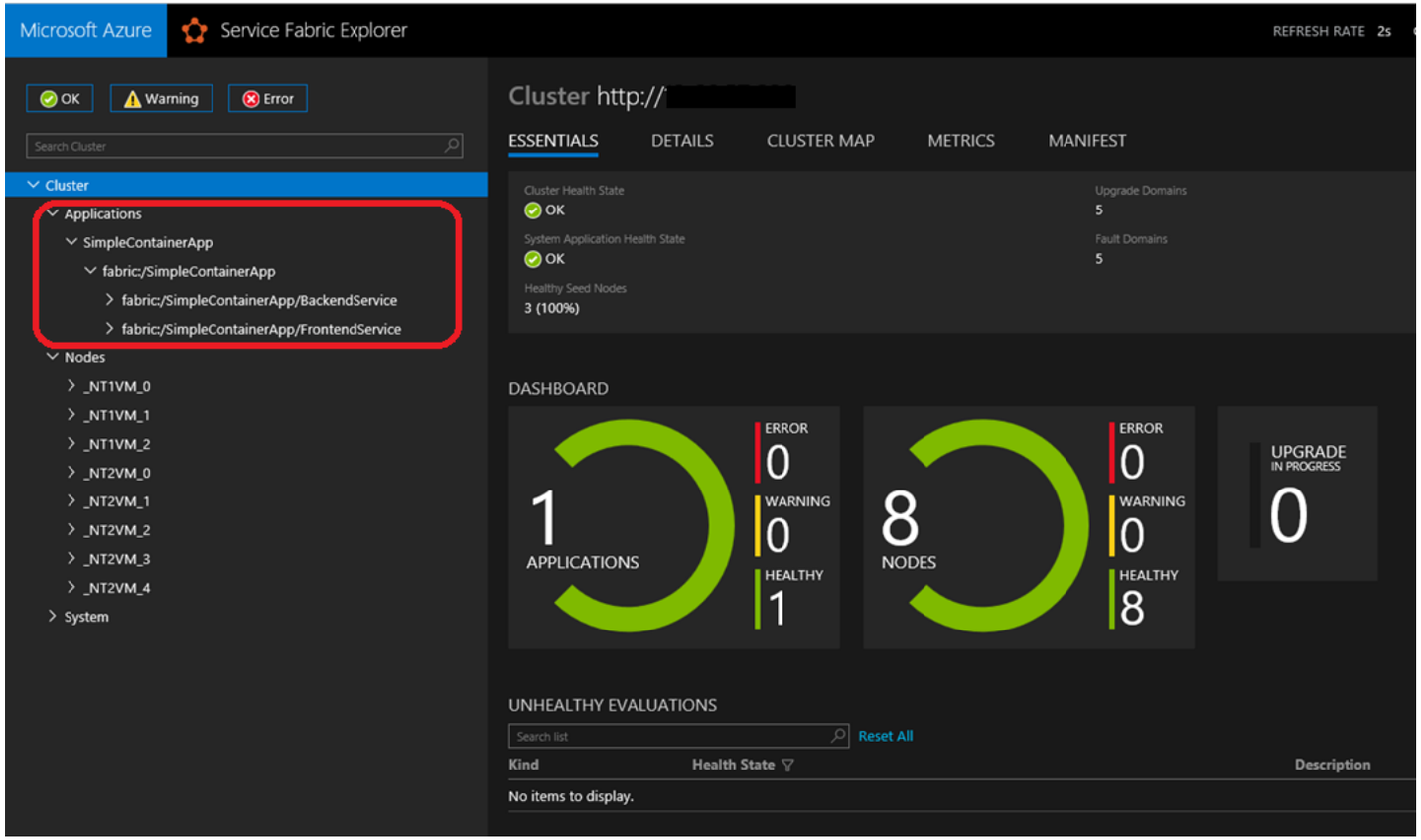
Microservices and Containers Orchestration Platform

Introduction to Azure Service Fabric

Microsoft Services

Azure Service Fabric

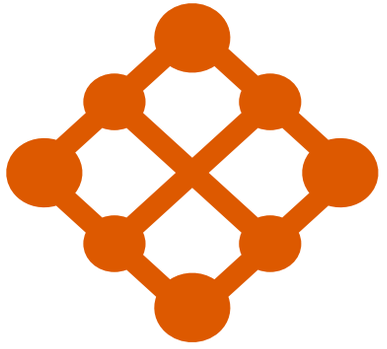
- Microservice and orchestration platform
- Build applications as containers **and/or** microservices
- Available on Windows & Linux
- Built-in cross-container communication
- Web based management UI
- Available On-Prem, Azure or other Clouds



Service Fabric

Windows and Linux Containers
Stateless and stateful microservices
Deploy on Azure, Azure Stack and on-premises

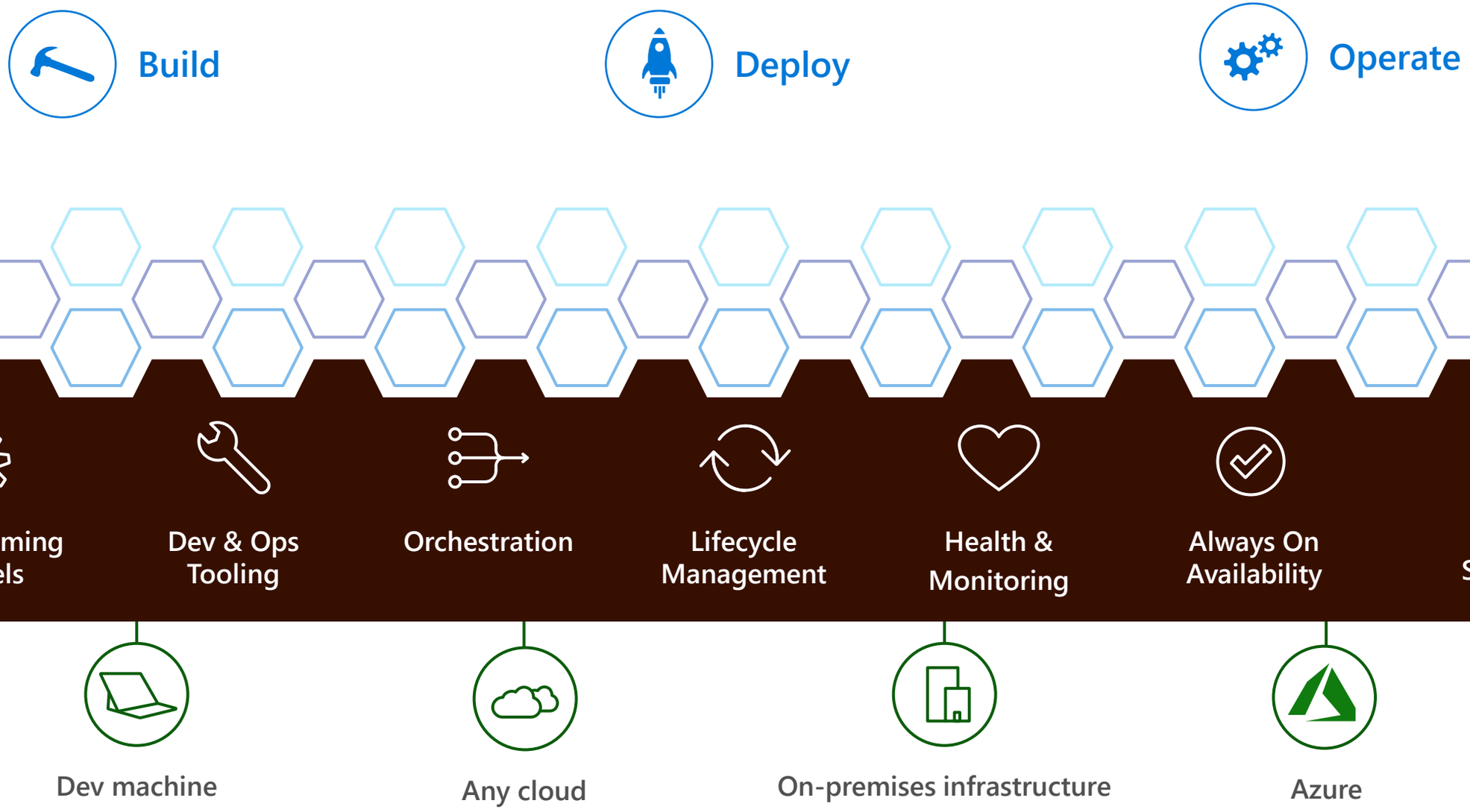
Service Fabric Mesh



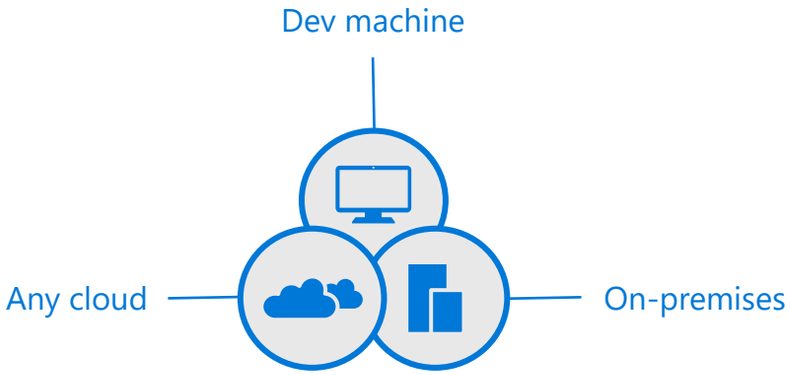
Fully managed microservices platform, built on Service Fabric
Windows and Linux Containers
Routing, data persistence and secrets management
Cost effective per second billing for all resources

Service Fabric: Cloud Application Platform

Build and deploy containers and microservices on Windows and Linux, at any scale, on any cloud



Azure Service Fabric offerings



Service Fabric Standalone

Bring your own infrastructure

Responsibility

You	Hardware OS patching Runtime upgrades Cluster capacity Network and storage App deployment
Azure	

Azure Service Fabric

Dedicated Azure clusters

You	Cluster capacity Network and storage App deployment
Azure	Virtual machines OS patching Runtime upgrades

Azure Service Fabric Mesh

Serverless microservices

You	App deployment
Azure	Virtual machines OS patching Runtime upgrades Capacity planning Network and storage Micro-billing

Microsoft

Microservices and Containers Orchestration Platform

Introduction to Kubernetes

Microsoft Services

Kubernetes

Kubernetes was created by Google and is one of the most feature-rich and widely used orchestration frameworks; its key features include:

- Automated deployment and replication of containers
- Online scale-in or scale-out of container clusters
- Load balancing over groups of containers
- Rolling upgrades of application containers
- Resilience, with automated rescheduling of failed containers
- Controlled exposure of network ports to systems outside of the cluster

Kubernetes: the industry leading orchestrator

Portable

Public, private, hybrid, multi-cloud

Extensible

Modular, pluggable, hookable, composable

Self-healing

Auto-placement, auto-restart, auto-replication, auto-scaling

Kubernetes

Deploy your applications quickly and predictably

Scale your applications on the fly

Roll out new features seamlessly

Limit hardware usage to required resources only

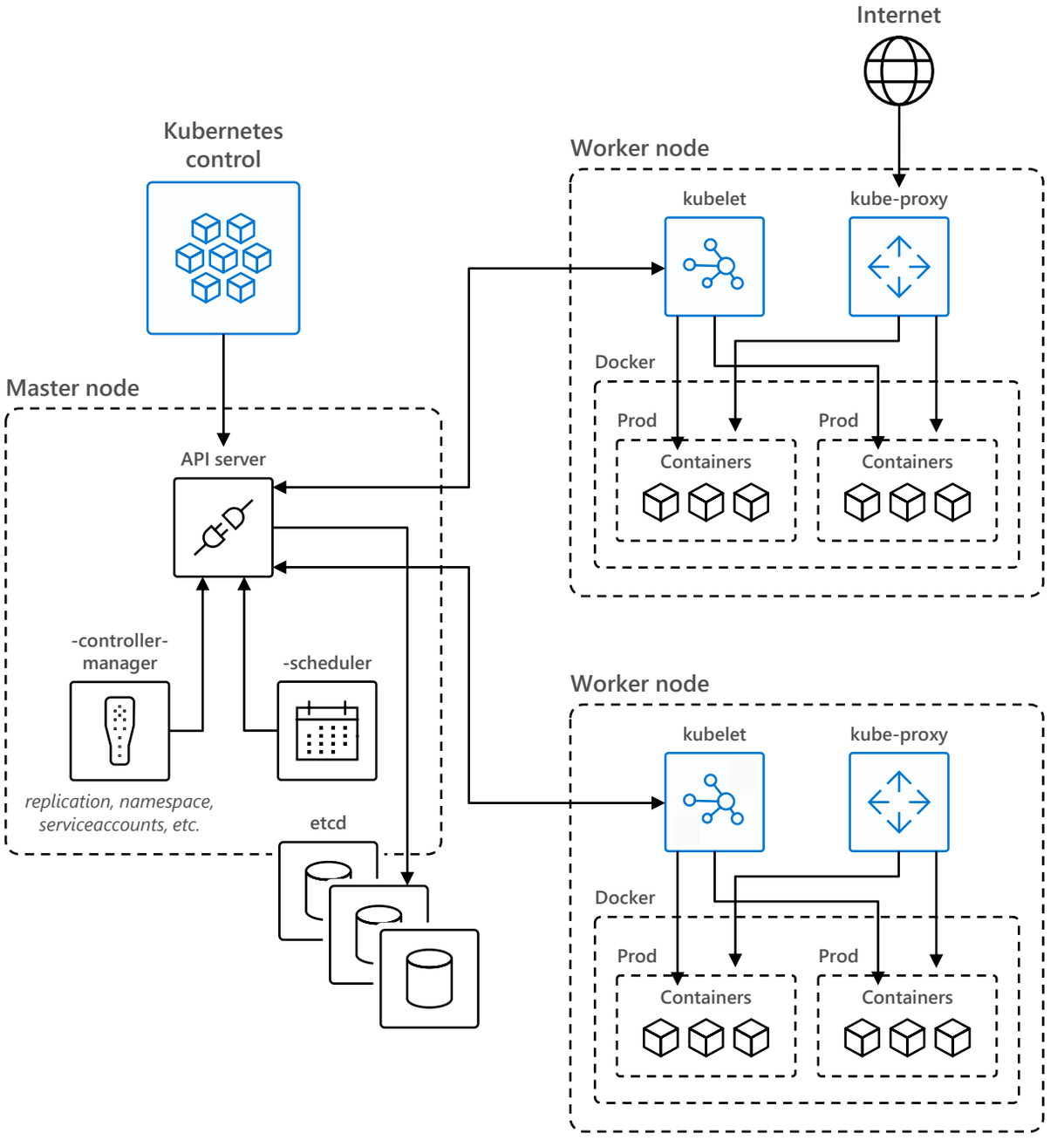
Kubernetes 101

1. Kubernetes users communicate with API server and apply desired state

2. Master nodes actively enforce desired state on worker nodes

3. Worker nodes support communication between containers

4. Worker nodes support communication from the Internet



Knowledge Check

- What are some of the main container orchestration platforms?
- How do microservices and containers differ?
- Why is container orchestration needed?