Microsoft

Modernizing Applications with Containers and Orchestrators

Microsoft Services

Microsoft

Module 5 – Container Orchestrators

Microsoft Services

Agenda

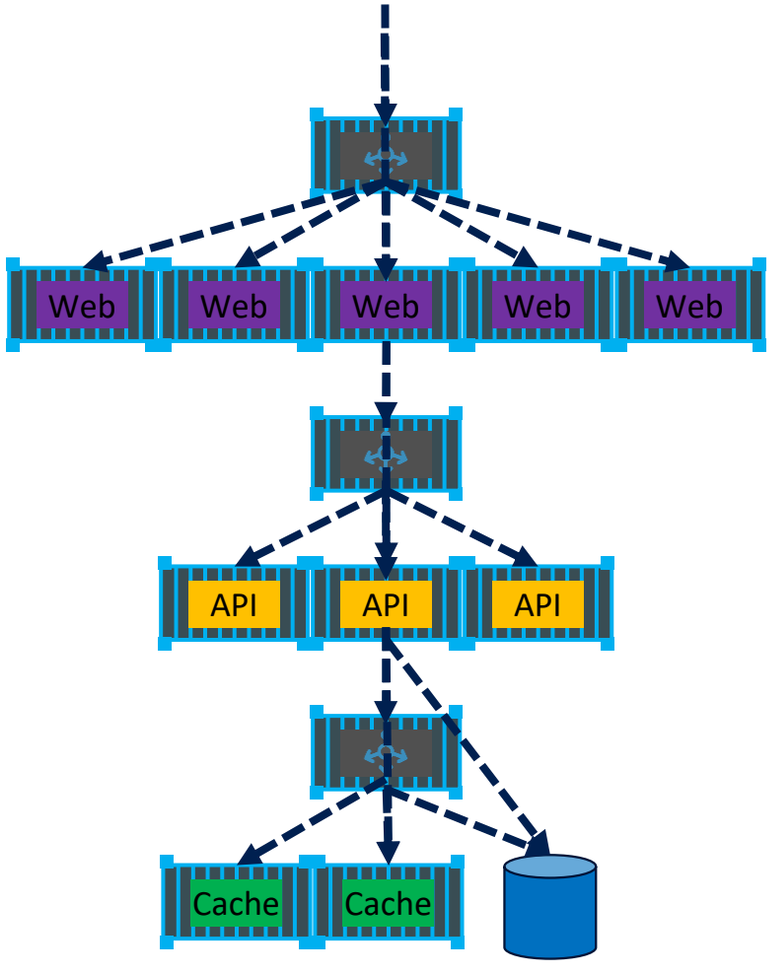
- What is orchestration?
- Microsoft offerings for containers
- Azure Container Instance
- Service Fabric
- Additional options for orchestration
- Introduction to Kubernetes
- Introduction to MiniKube

Challenges of a containerized world

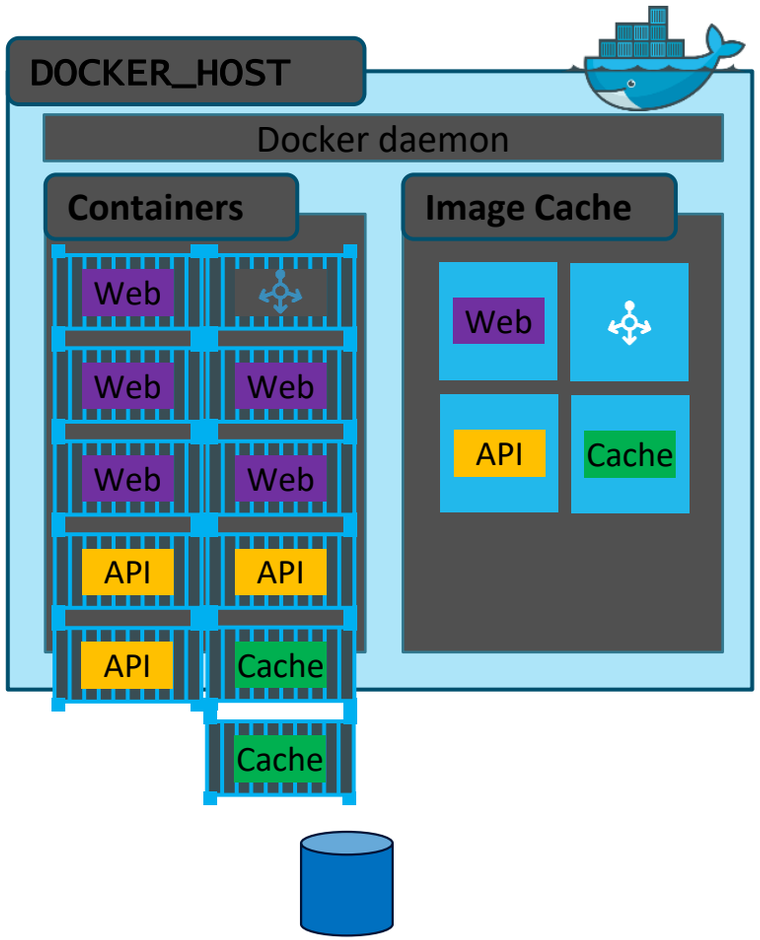
As application development has moved towards a container-based approach, the need to orchestrate and manage the inter-connected resources becomes important

- Load Balancing
  - Distributing traffic across containers at scale
- Naming and Discovery
  - How do containers or groups find one another?
- Logging and Monitoring
  - Keeping track of what containers are doing
- Debugging and Introspection
  - Getting inside running containers
- Networking
  - Differentiating container networks from host networks at scale

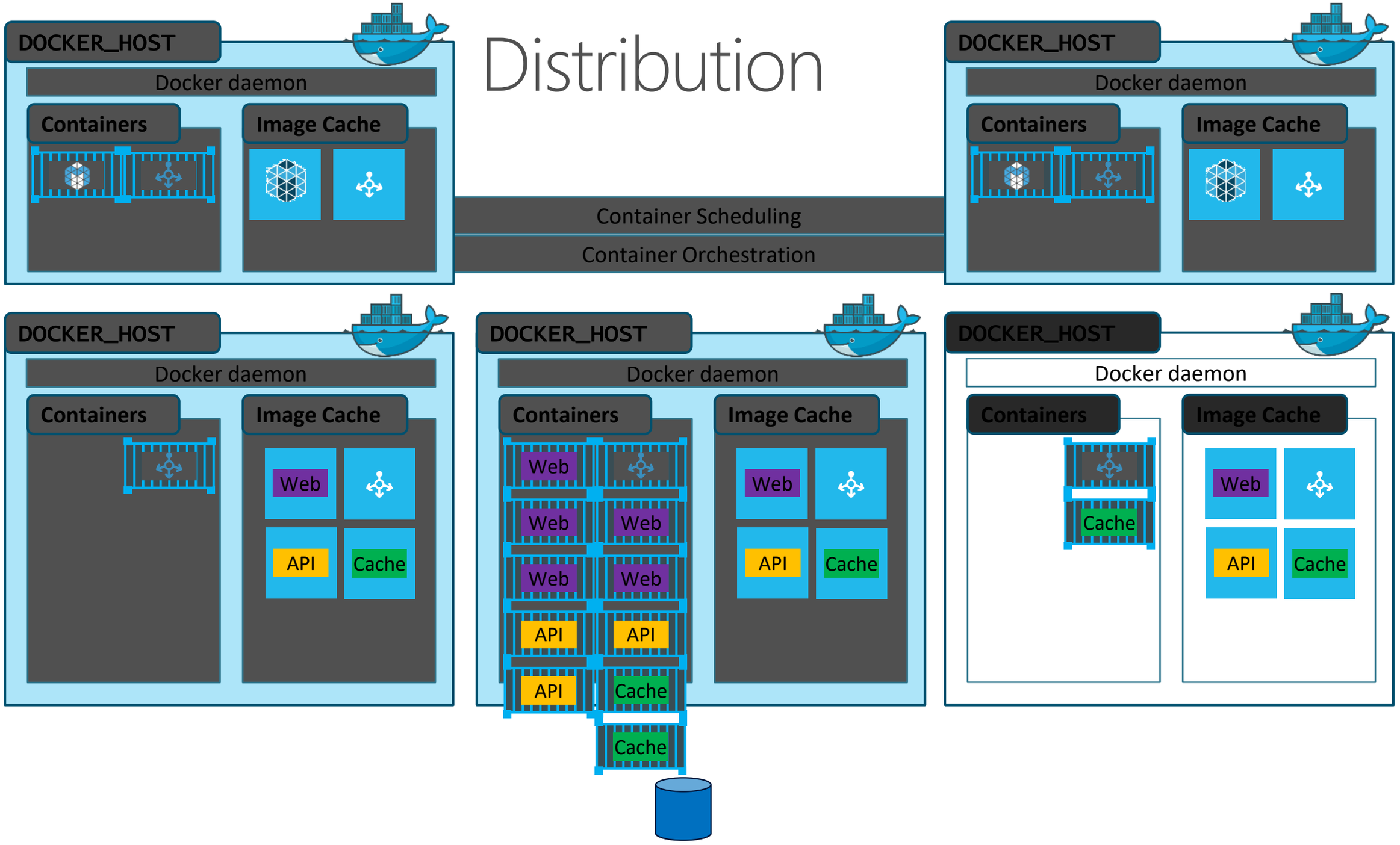
Application Scale



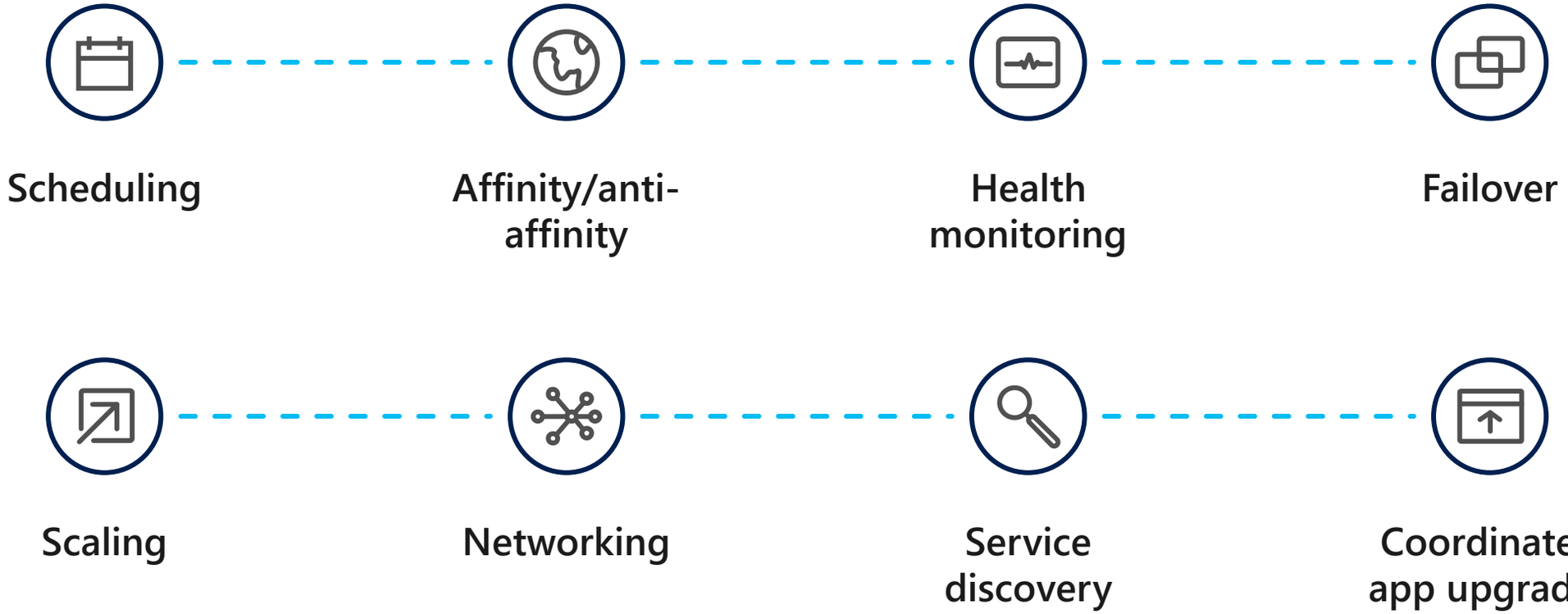
Load Balancing & Fault Tolerance



Distribution



The elements of orchestration



Clustering versus orchestration

Clustering

- Grouping “hosts”—either VMs or bare metal—and networking them together
- A cluster should feel like a single resource rather than a group of disparate machines

Orchestration

- Managing and monitoring of the workloads running in your cluster
- Starting containers on appropriate hosts and connecting them
- May also include support for scaling, automatic failover, and node rebalancing

Microsoft Offerings for Containers

IF YOU'RE LOOKING FOR THIS...	USE THIS
Scale and orchestrate containers using Kubernetes	Kubernetes Service
Easily run containers on Azure with a single command	Container Instances
Store and manage container images across all types of Azure deployments	Container Registry
Develop microservices and orchestrate containers on Windows or Linux	Service Fabric
Deploy web applications on Linux using containers	App Service
Run repetitive compute jobs using containers	Batch

Azure App Service

Easily deploy and run container-based web apps at scale

Accelerated outer loop

Tight integration w/ Docker Hub, Azure Container Registry

Built-in CI/CD w/ Deployment Slots

Intelligent diagnostics & troubleshooting, remote debugging

Fully managed platform

Automatic scaling and load balancing

High availability w/ auto-patching

Backup & recovery

Flexibility & choices

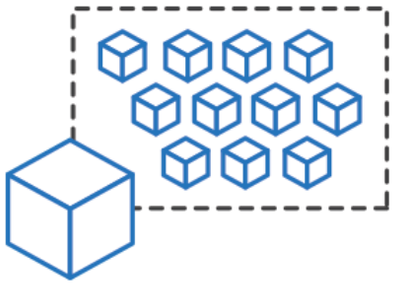
From CLI, portal, or ARM template

Single Docker image, multi container w/ Docker Compose

IntelliJ, , Jenkin, Maven Visual Studio family

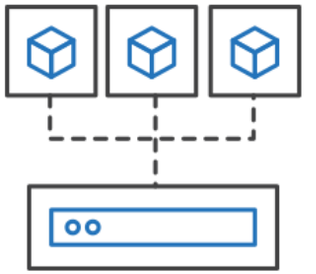
Azure Container Instances (ACI)

Easily run containers on Azure without managing servers



Run containers without managing servers

Increase agility with containers on demand



Secure applications with hypervisor isolation

Azure Container Instances

Great for:

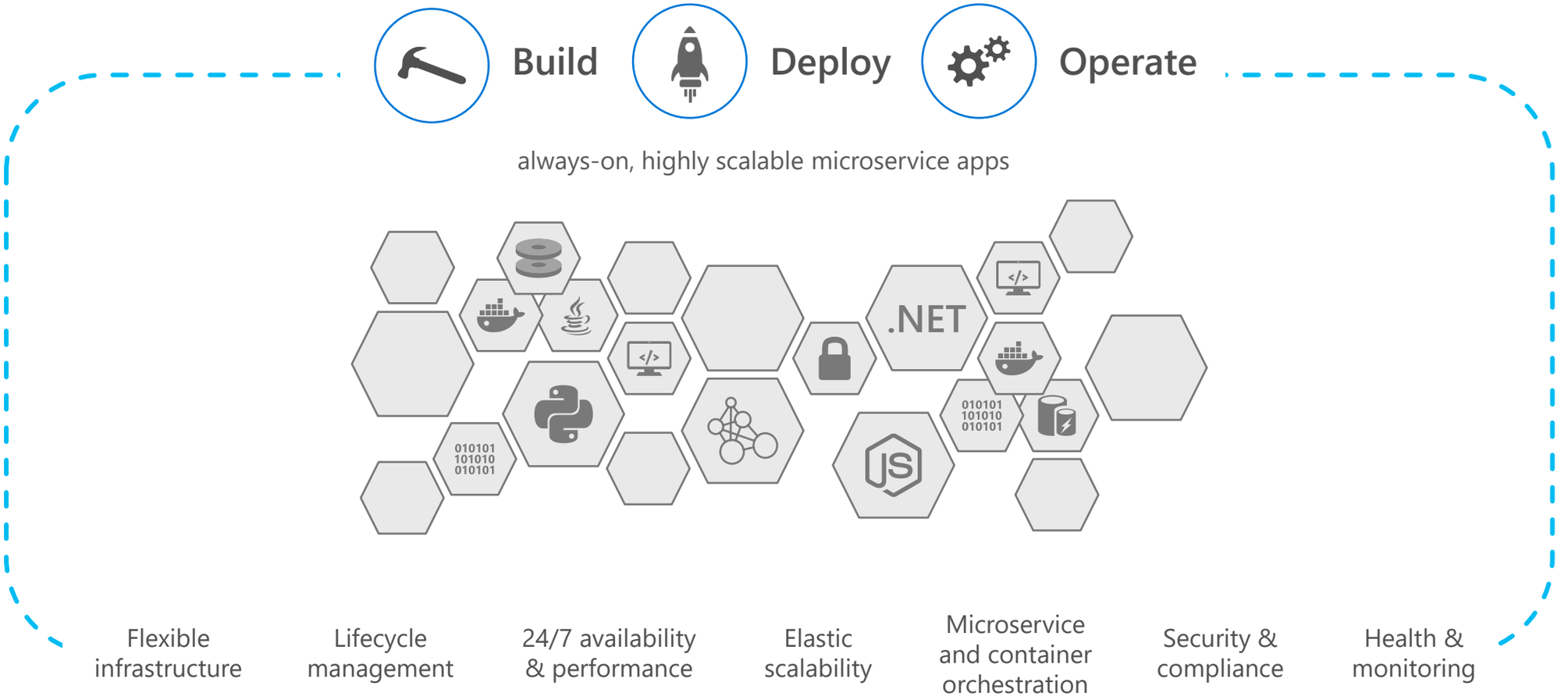
- Isolated Windows and Linux containers
- Simple applications
- Task automation
- Build jobs
- Hypervisor-level security
- Custom sizes for CPU cores and memory
- Public IP connectivity
- Persistent storage
- Co-scheduled groups

NOT great for:

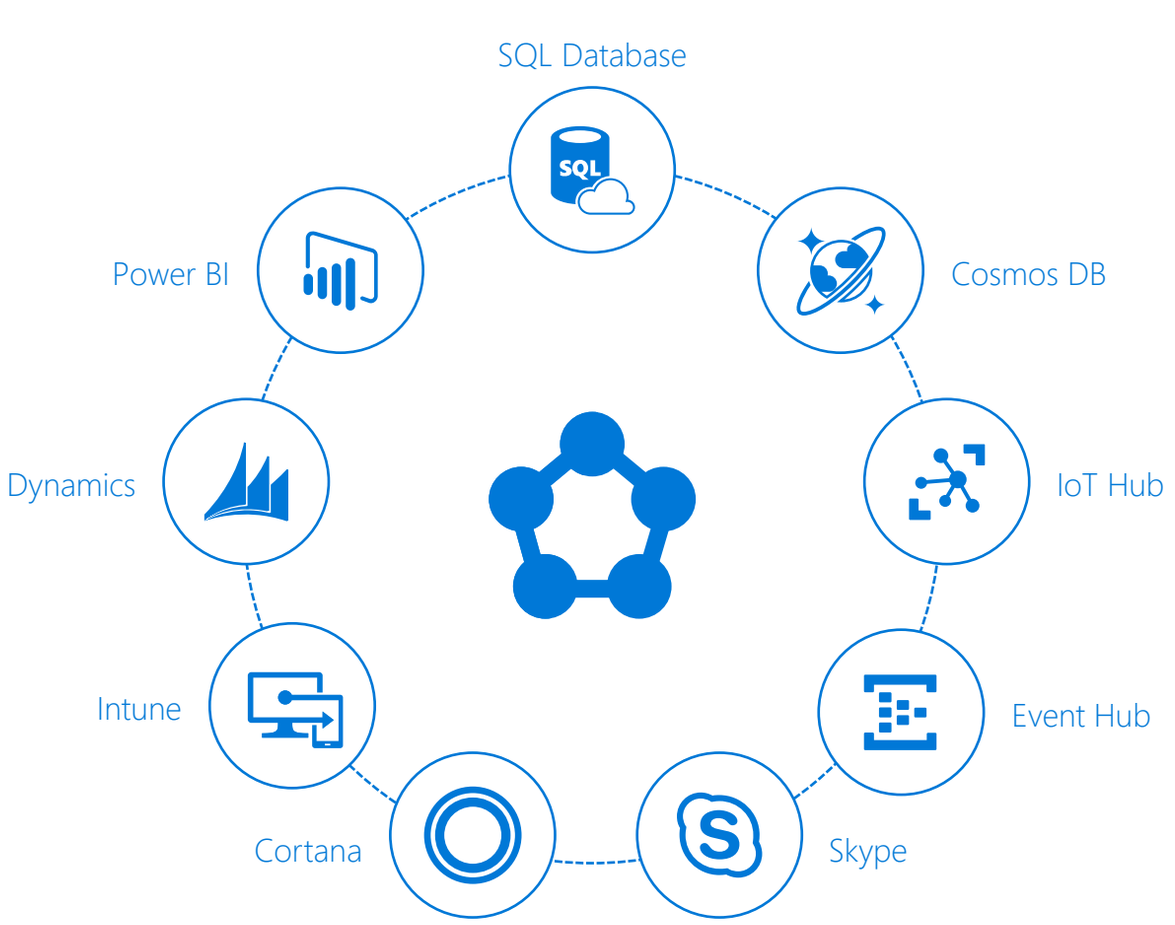
- Full container orchestration
- Service discovery across multiple containers
- Automatic scaling
- Coordinated application upgrades
- For all of above Microsoft recommend the Azure Kubernetes Service.

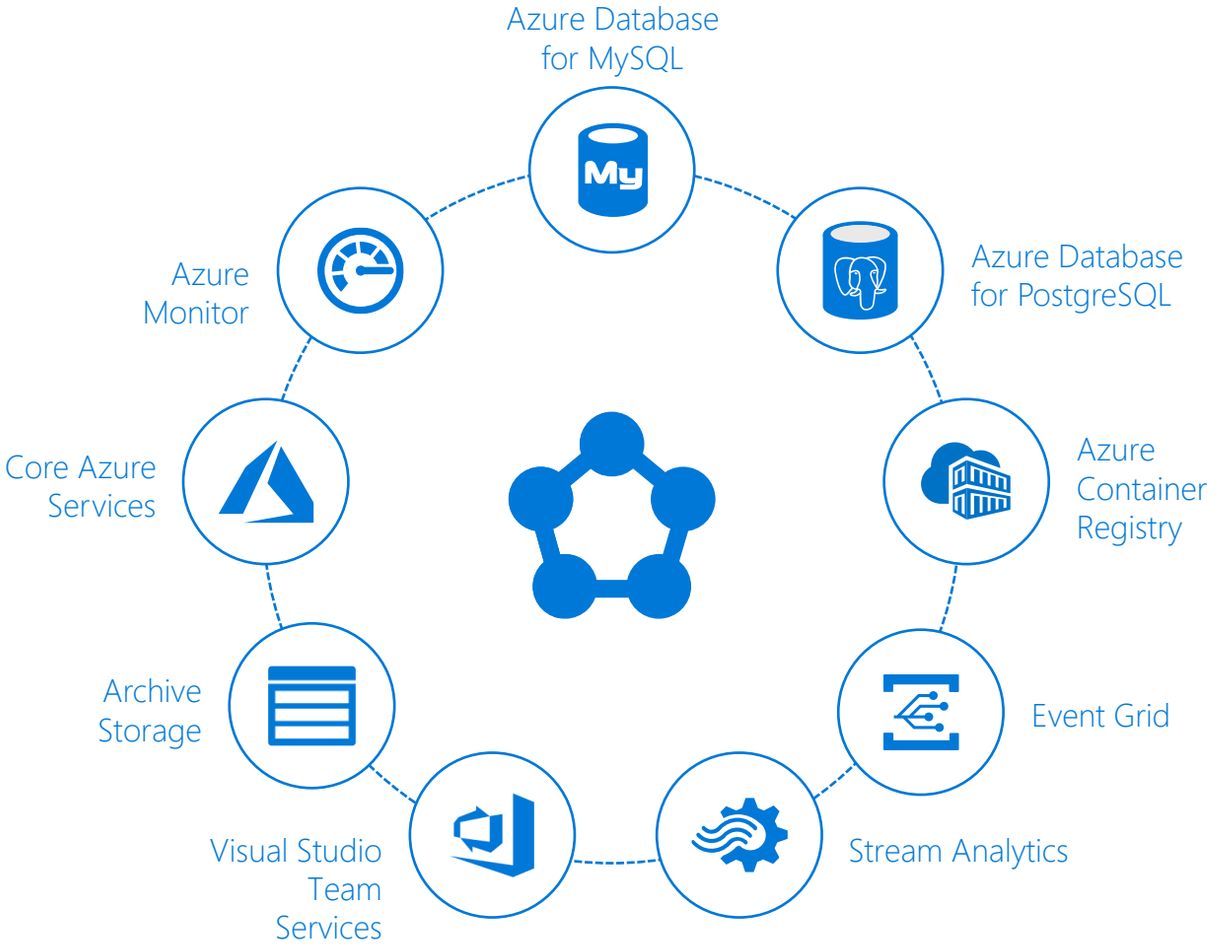
Azure Service Fabric

A microservices platform for business critical applications



PaaS built on Service Fabric





Service Fabric capabilities

- Application deployment services:
  - Rolling update with rollback - Upgrade and patch microservices within applications independently
  - Strong versioning
  - Side-by-side support for same application type
- Leadership election
- Naming service for discovery of applications
- Partitioning support
- Azure Load balancing and placement constraints
- Consistent state replication framework
- Ability to scale-out or scale-in your Service Fabric cluster
- Now provides a serverless solution with Azure Service Fabric Mesh

Microsoft

Kubernetes

Overview of Kubernetes

Microsoft Services

Kubernetes

- **Kubernetes** is "an open-source software for automating deployment, scaling, and management of containerized applications“
- **Kubernetes**, in Greek κυβερνήτης, means the Helmsman, or pilot of the ship
- Keeping with the maritime theme of **Docker** containers, **Kubernetes** is the pilot of a ship of containers

Kubernetes

The de-facto orchestrator

Portable

Public

Private

Hybrid

multi-cloud

Extensible

Modular

pluggable

Hookable

composable

Self-healing

Auto-placement

auto-restart

auto-replication

auto-scaling

Kubernetes

Empowering you to do more

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 Terminology

Cluster

- Control Plane (Master Node) schedules containers
- Worker nodes run containers

Kubectl

- main command line tool to manage your cluster

Kubernetes Terminology

Pod – one or more containers. Smallest unit of deployment

ReplicaSet – Multiple instances of a pod

Deployment – defines desired state of Pods

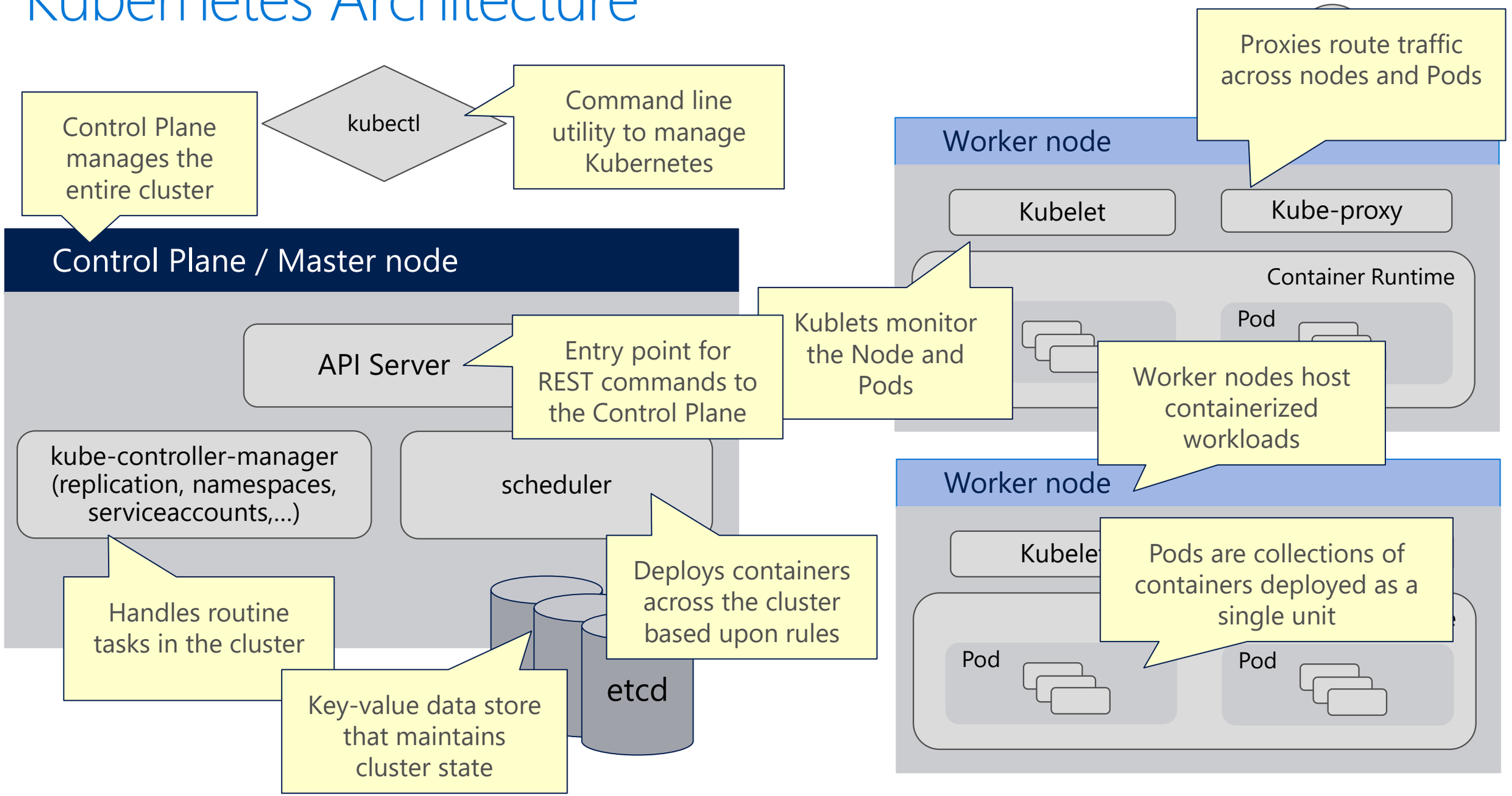
Service – Load Balancing. Locates Pods and routes traffic

Namespace – Isolation. Used to isolate applications, version or environments

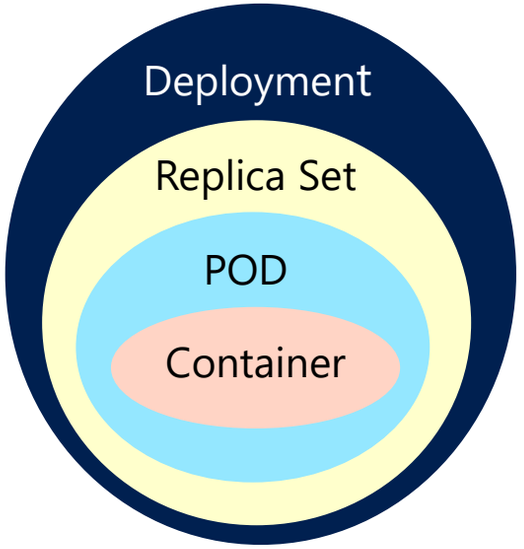
YAML – Declarative Deployments

HELM – package manager for kubernetes

Kubernetes Architecture



What are Pods?



- Pods are the smallest building block in Kubernetes…
  - **A collection of co-located containers and volumes**
  - **Running in the same execution environment**
  - **Managed as a single atomic unit**
- You never directly run a container, instead you run a Pod
- Apps running in a Pod share the same IP, port and communicate using native interprocess communication channels
- Apps in different Pods are isolated from each other; they have different IP addresses, different hostnames, etc.
- Pods are immutable - if a change is made to a pod definition, a new pod is created, and the old pod is deleted

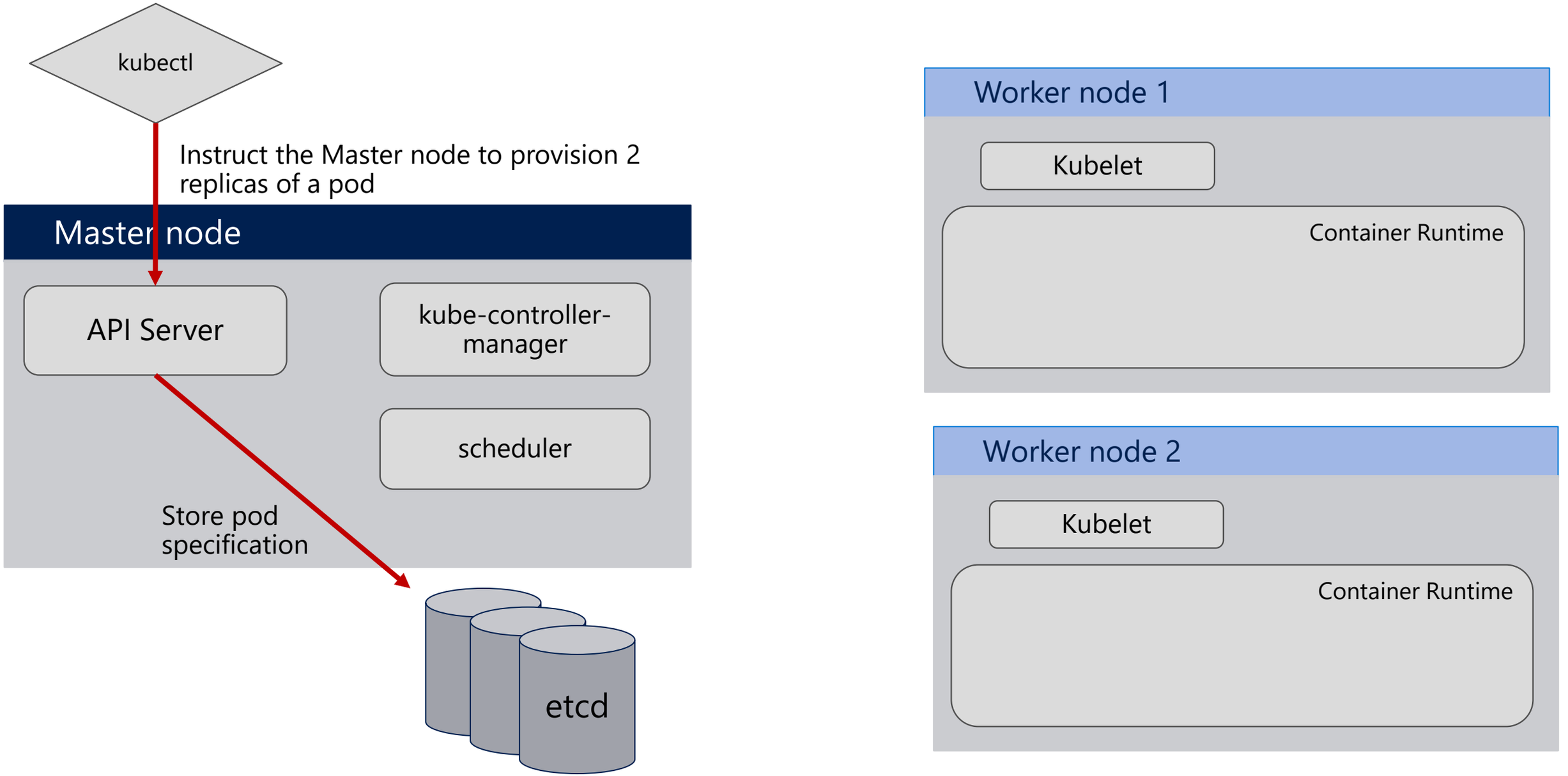
Declarative Configuration

- Pods are defined in a Pod manifest: A readable, declarative text-file
- Kubernetes itself thrives on declarative configuration…
  - Capture the desired state of a Kubernetes object in a configuration
  - Submit that configuration to a service that takes actions to ensure the desired state becomes the actual state
  - Provides for a more manageable, dynamic and reliable system
- Contrast with imperative configuration where you explicitly instruct the system what to do, typically by issuing a series of commands

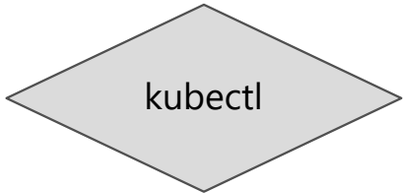
The Pod Lifecycle

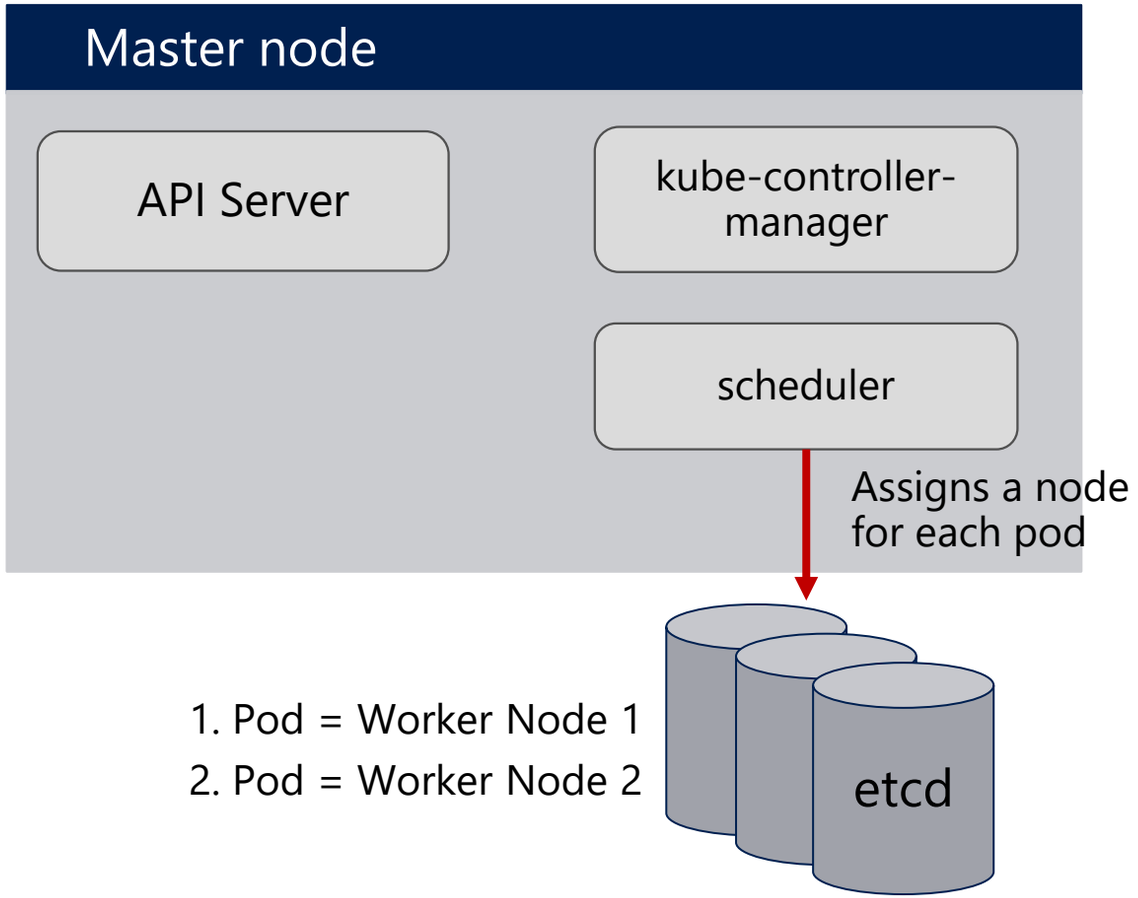
- Let's say you want to provision a container in Kubernetes
- From the Kubectl console, you…
  - Make a Pod request to an API server using a Pod definition (YAML) file
  - The API server saves the configuration data to the persistent storage (ETCD store)
  - The scheduler finds the unscheduled Pod and schedules it to an available node
  - The Kubelet sees the Pod scheduled and fires up Docker
  - Docker runs the container
- The Kubelet manages objects on the worker nodes
- The entire lifecycle state of the Pod is stored in the Etcd store

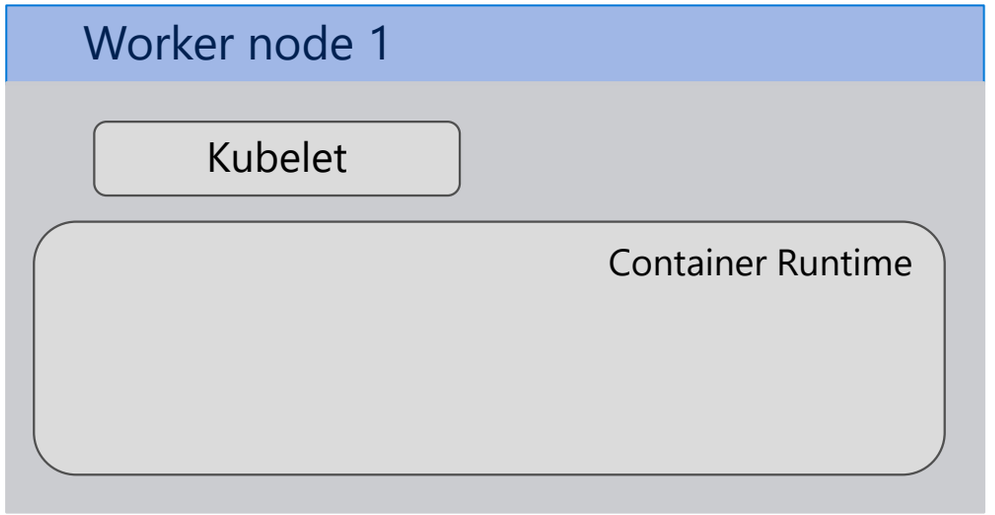
How Pods Work

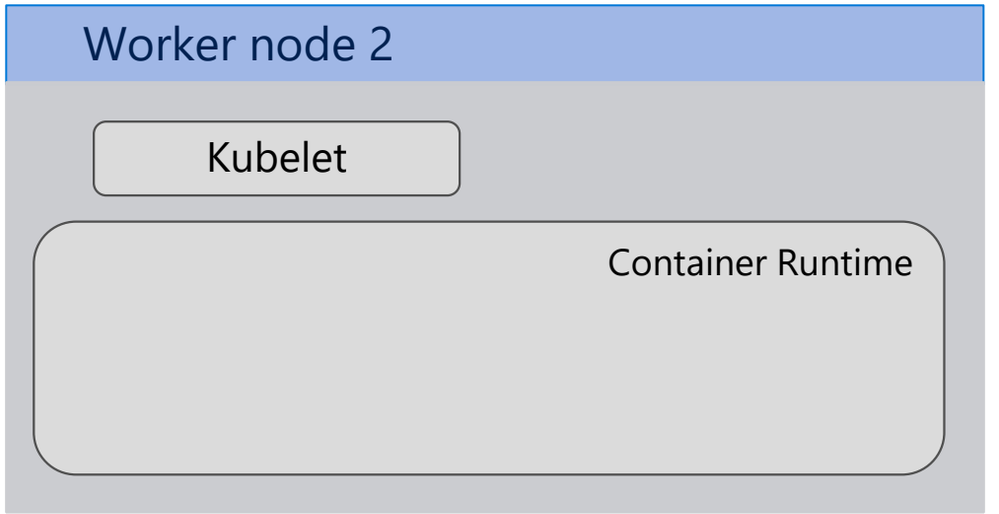


How Pods Work

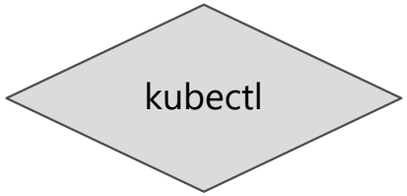


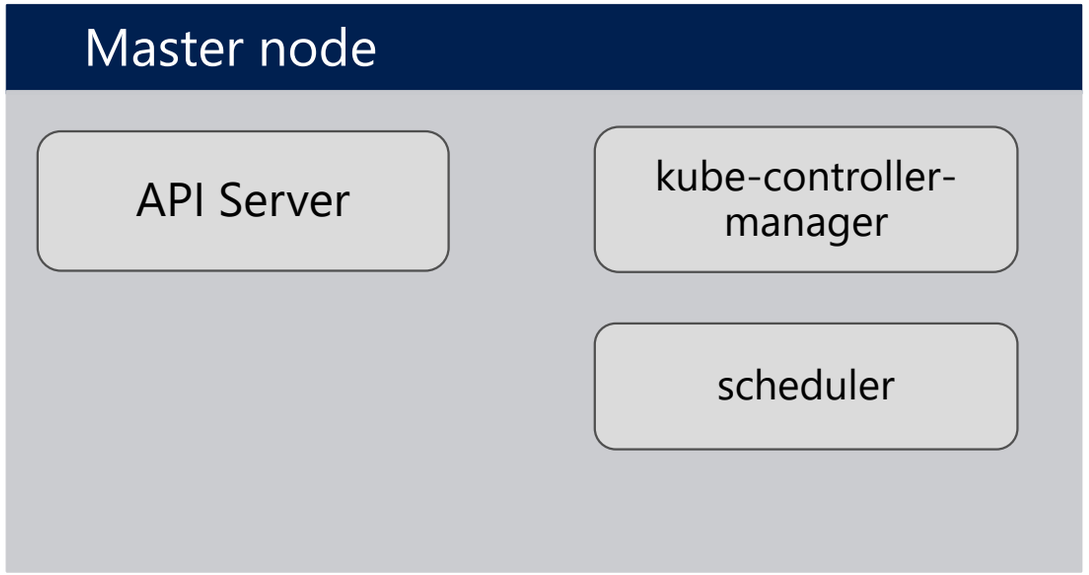




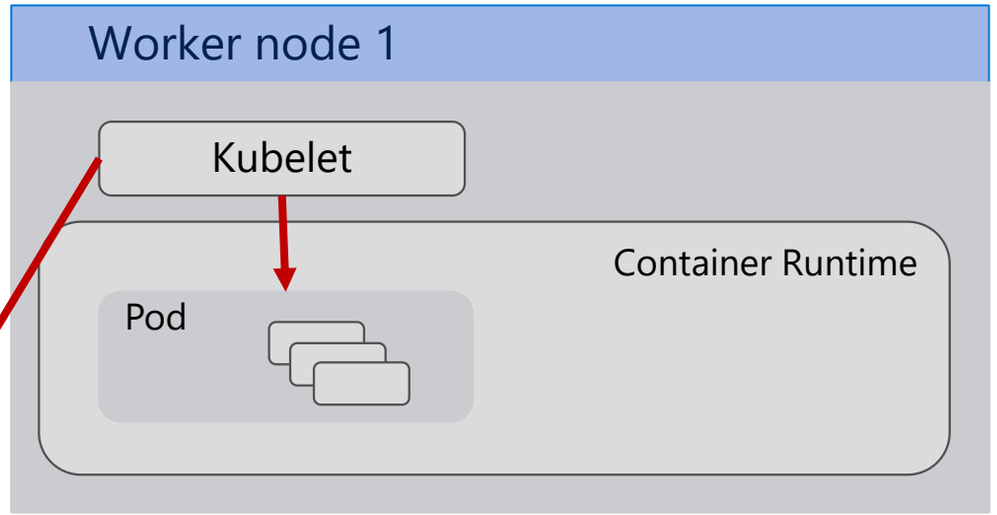


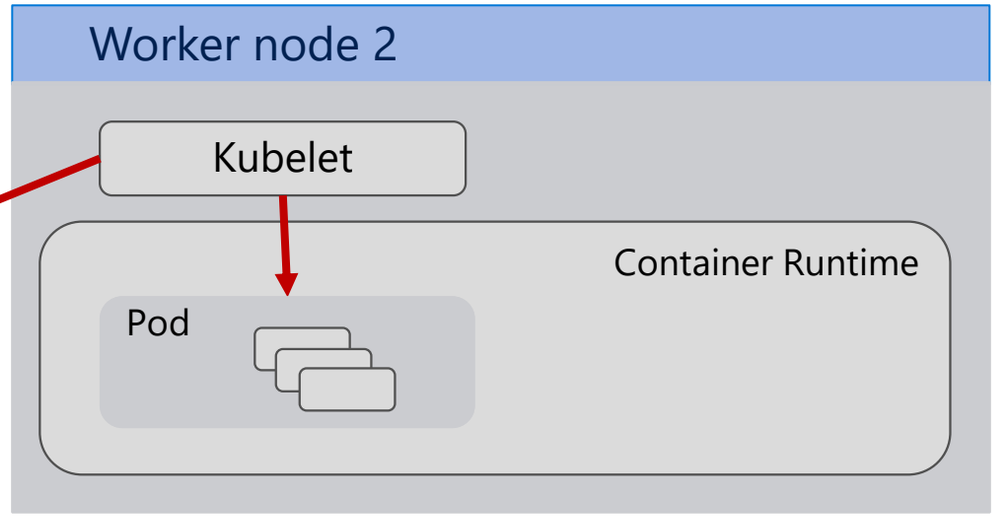
How Pods Work



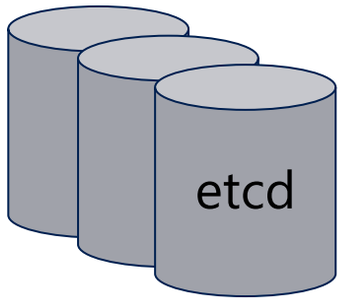


The Kubelets check if there are pods assigned and starts them

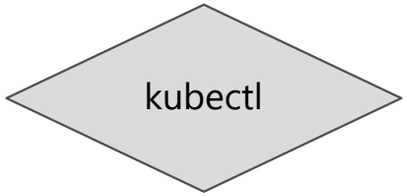




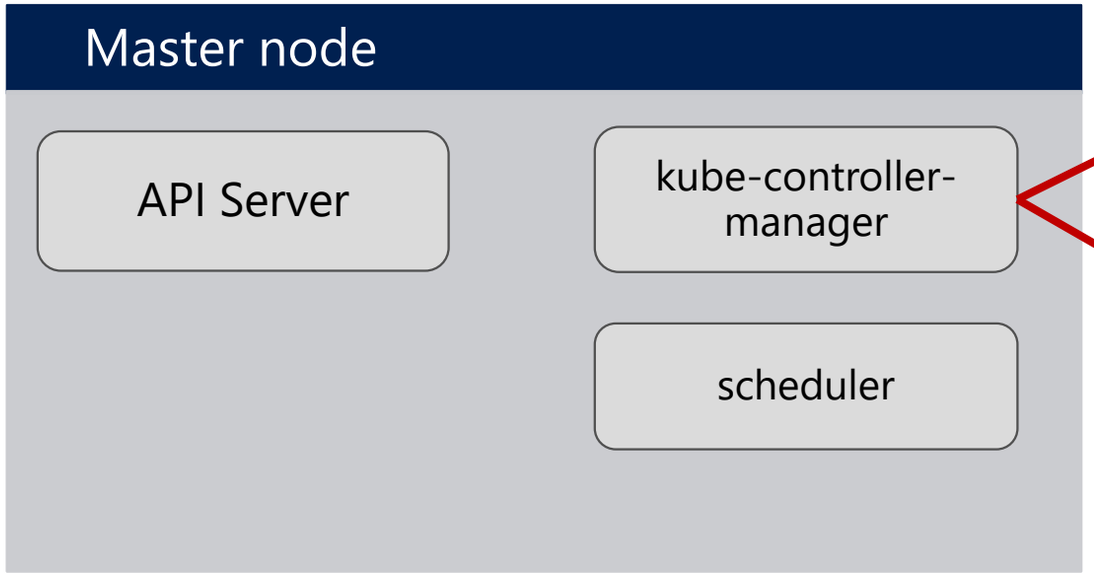
1. Pod = Worker Node 1
2. Pod = Worker Node 2

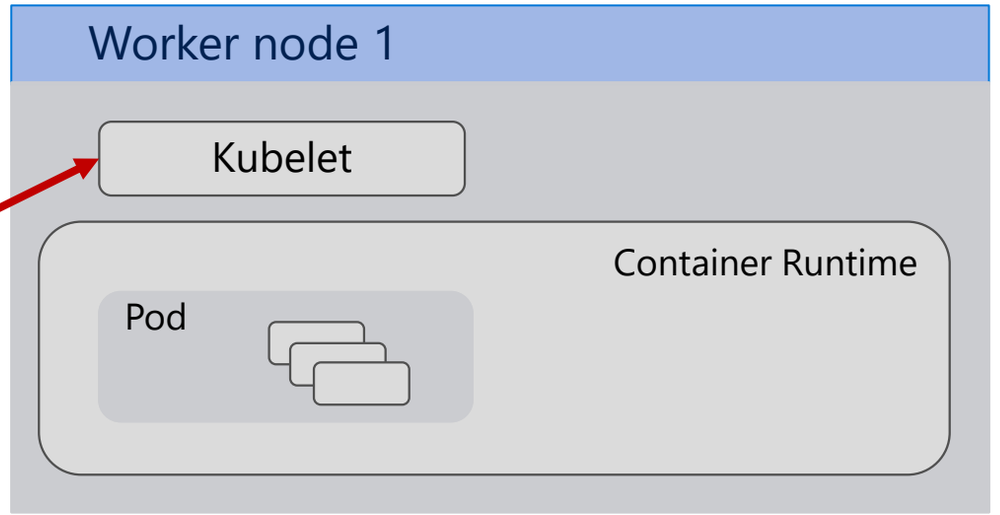


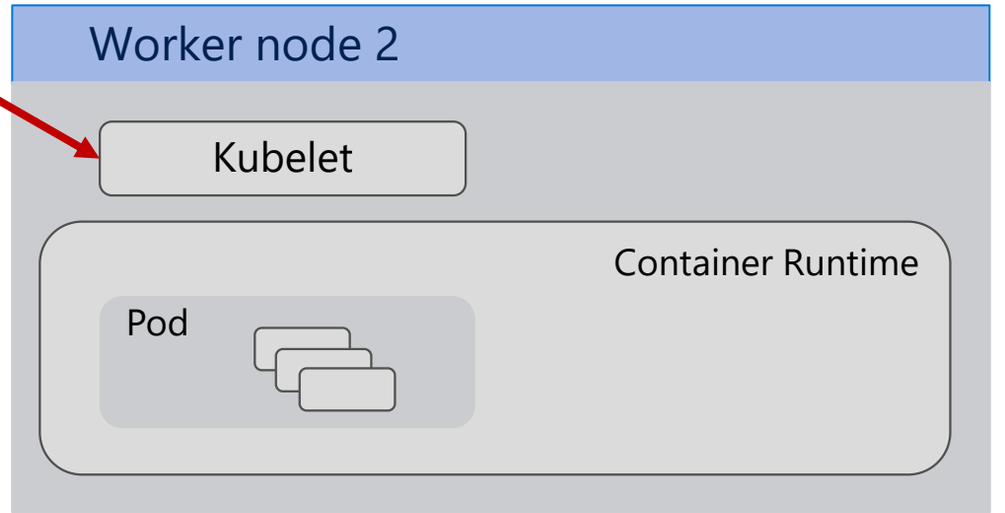
How Pods Work



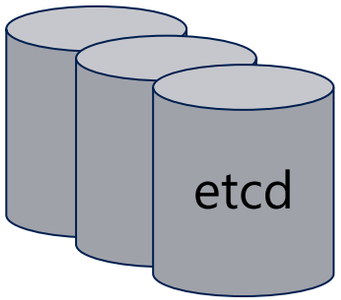
Kube-controller-manager ensures the correct number of pods is running in the cluster



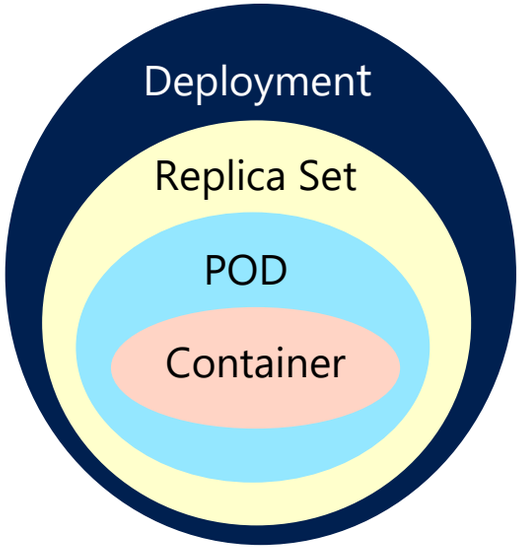




1. Pod = Worker Node 1
2. Pod = Worker Node 2

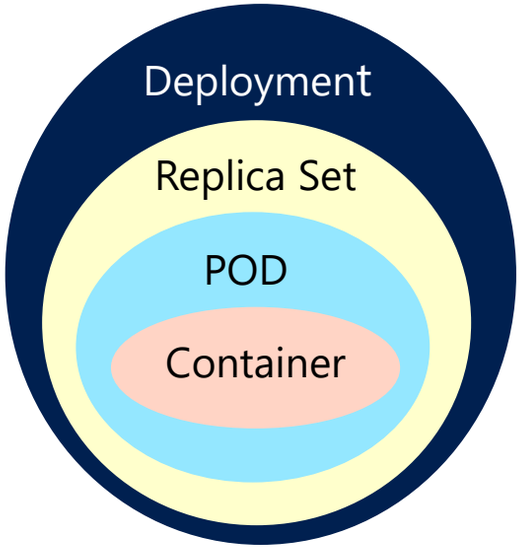


What are Replica Sets?



- A Pod is essentially a one-off singleton instance
- Replica Sets are a Kubernetes object that manage Pods
  - Redundancy – allow for failure
  - Scale – allow for more requests to be processed
- They monitor the cluster and ensure the desired number of Pods are correctly running
  - If no Pods are provisioned, the Replica Set Controller will schedule them
  - If actual count drops below the desired, the controller will schedule replacements
  - If you exceed the desired count, the controller will destroy them
- Replica sets are created by and managed through Kubernetes Deployment objects

What are Deployments?



- A deployment defines the lifecycle of an application
  - Is made up of pods
  - Controls Replica Sets
  - Includes the functionality to update the desired state
  - Rolling updates are included
  - Provides fine-grained control over how and when a new pod version is rolled out as well as rolled back to a previous state

- With a deployment, you can declaratively state how many instances of your pod you would like, you can define rollout strategies, gain self-healing behavior, and much more. This provides a scalable platform to deploy your application.

Azure Kubernetes Engine

Open source: https://github.com/Azure/aks-engine

- Easiest way to provision a self-managed Kubernetes cluster on Azure
- Leverages Azure Resource Manager (ARM), to help you create, destroy and maintain clusters provisioned with basic IaaS resources in Azure
- Allows you to customize Deployments
  - Deploying into existing virtual networks
  - Utilizing multiple agent pools

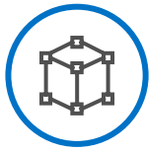
Azure Kubernetes Service

Simplify the deployment, management, and operations of Kubernetes

Deploy and manage Kubernetes with ease

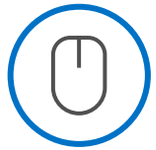
Scale and run applications with confidence

Secure your Kubernetes environment



Accelerate containerized application development

Work how you want with open-source tools & APIs



Set up CI/CD in a few clicks

Why Azure Kubernetes Service?

Easy to use

- Fastest path to Kubernetes (K8s) on Azure
- Up and running with 3 simple commands
- AKS is free - you only pay for the agent nodes within your clusters, not for the masters

Easy to manage

- The K8s masters are managed by Azure
- Automated upgrades and patching
- Easily scale the cluster up and down
- Self-healing control plane

Uses Open APIs

- 100% upstream Kubernetes

AKS Compliance and Certification

- Azure Kubernetes Service (AKS) has been CNCF certified as Kubernetes conformant.
- Azure Kubernetes Service (AKS) is compliant with SOC, ISO, and PCI DSS.

Demonstration: *Kubernetes Cluster in Azure Kubernetes Service*

Deploy Kubernetes clusters in Azure Kubernetes Service

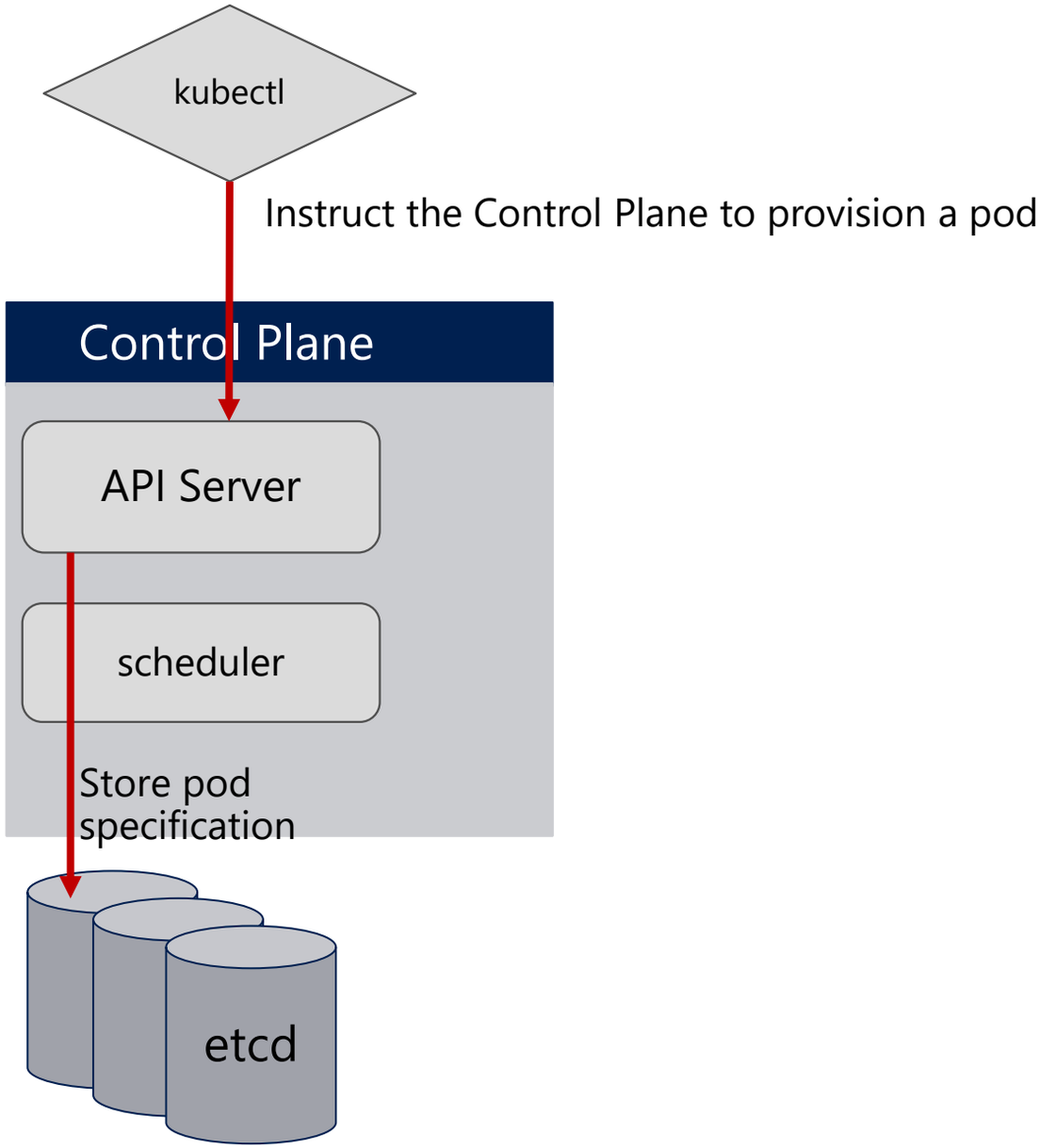
Deploy NGINX container into Kubernetes cluster

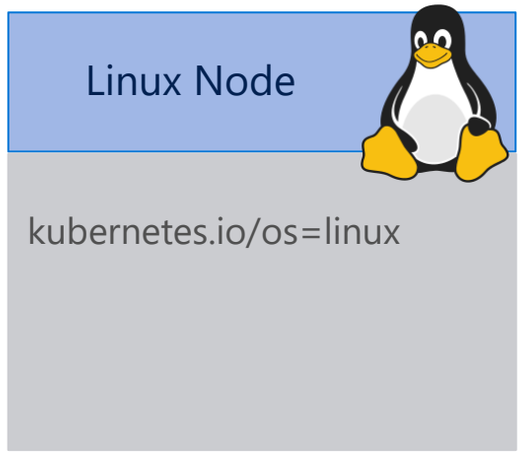
Azure Kubernetes Service

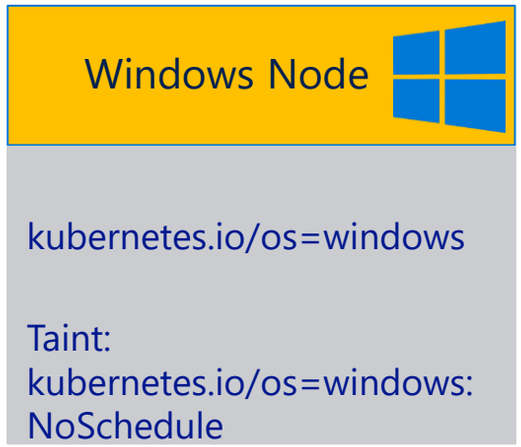
Windows Support

- AKS clusters with Windows node pools must use the Azure CNI (advanced) networking model
- AKS cluster supporting Windows containers will be composed of a Windows node pool and a Linux node pool
- Use Node **Taints** to prevent Linux pods from deploying to Windows Nodes.
- Use **Node Selector** to specify on which nodes the workloads must be deployed

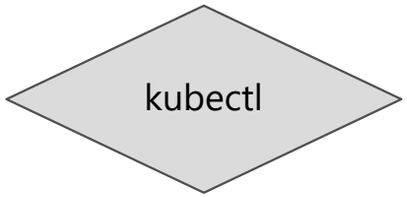
Deploying Windows Pods

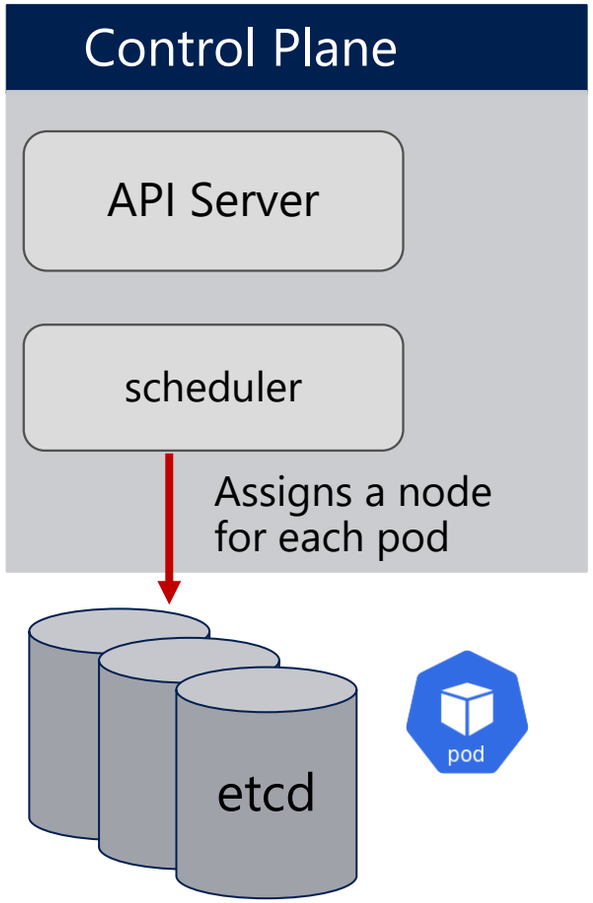






Deploying Windows Pods





Pod Manifest

```
spec:
  tolerations:
   - key: kubernetes.io/os
     operator: Equal
     value: windows
     effect: NoSchedule
  nodeSelector:
     kubernetes.io/os: windows
```

A Pods Tolerations informs Kubernetes to include Nodes with the same Taint

A Pods Node Selector informs Kubernetes to **only** include Nodes with the same Label

Windows Node

kubernetes.io/os=windows

Taint:
kubernetes.io/os=windows
NoSchedule

The Node Taint prevents the node from being considered

Demonstration: *Windows Containers*

Deploy a Windows Container

Minikube for local k8s development

Runs a single-node Kubernetes cluster inside a VM on your laptop, allowing you to try out Kubernetes or develop with it day-to-day

Supports Kubernetes features such as:

- DNS
- NodePorts
- ConfigMaps and Secrets
- Dashboards
- Container Runtime: Docker, rkt and CRI-O
- Enabling CNI (Container Network Interface)
- Ingress

Docker + k8s integration

- Since early 2018, Docker platform integrates with Kubernetes
- Developers and operators can build apps with Docker and seamlessly test and deploy them using both Docker Swarm and Kubernetes
- Kubernetes support comes with both Docker Enterprise Edition (EE) & Docker Community Edition (CE)

Demonstration: *Working with Kubernetes Minikube*

Minikube Overview

Working with Deployments, and Replica Set

Labs

Lab 5: Orchestrators (windows)

Lab 5: MiniKube (linux)