Container Orchestrators

Robert Rozas Navarro
Customer Engineer
Apps-Dev Domain

Microsoft

Objectives

- Understanding Role of Orchestration
- Azure Kubernetes Service (AKS)
- Azure Container Service (Kubernetes, Swarm, DC/OS)
- Azure Container Registry
- Azure Container Instances
- Demos

Why Containers Needs Orchestration?

Containers have a need:

- to discover and talk to each other
- to manage state
- to be upgraded with zero down time
- to report health & resource usage
- to be placed appropriately
- to be scaled in/out on demand
- to be resilient to HW and SW faults

Orchestration

Orchestration is:

- Automatic Failover
- Monitored Upgrades
- State Management
- Resource Monitoring and Management
- Constraints
- Health Checks

More…

Orchestration versus Clustering

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

- Making all the pieces work together. Starting containers on appropriate hosts and connecting them. An orchestration system may also include support for scaling, automatic failover, and node rebalancing.

Microsoft Various Offerings for Containers

IF YOU'RE LOOKING FOR THIS...	USE THIS
Scale and orchestrate containers using Kubernetes, DC/OS or Docker Swarm	Container 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

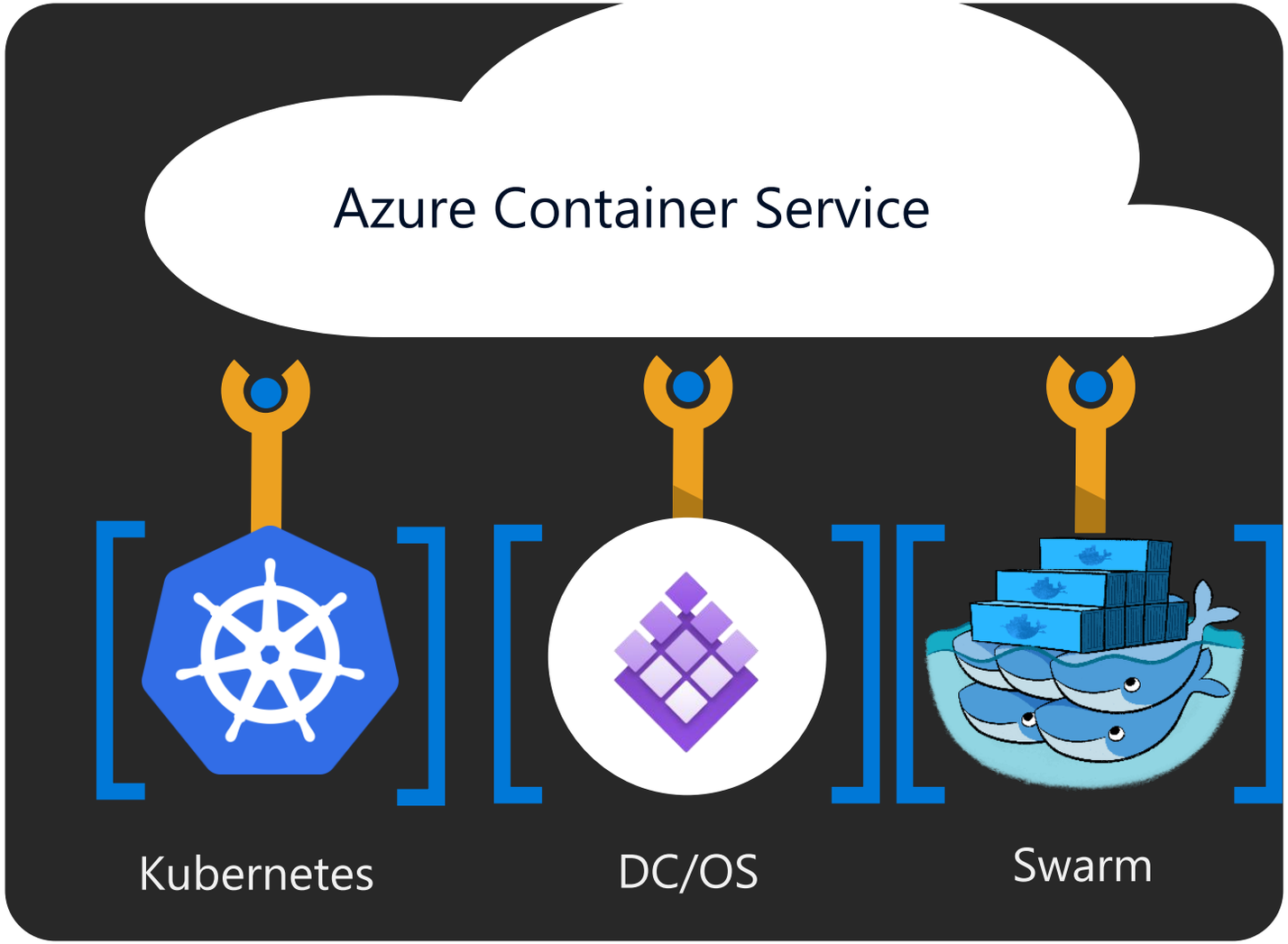
Azure Kubernetes Service (AKS)

- Simplify Kubernetes management, deployment, and operations.
- Easily provision clusters via the Azure portal and Azure CLI, or with infrastructure as code tools such as Azure Resources Manager and Terraform. Simplify cluster maintenance with automated upgrades and scaling.
- Gain operational visibility into your managed Kubernetes environment with control plane telemetry, log aggregation, and container health visible as part of the Azure portal, automatically configured for AKS clusters

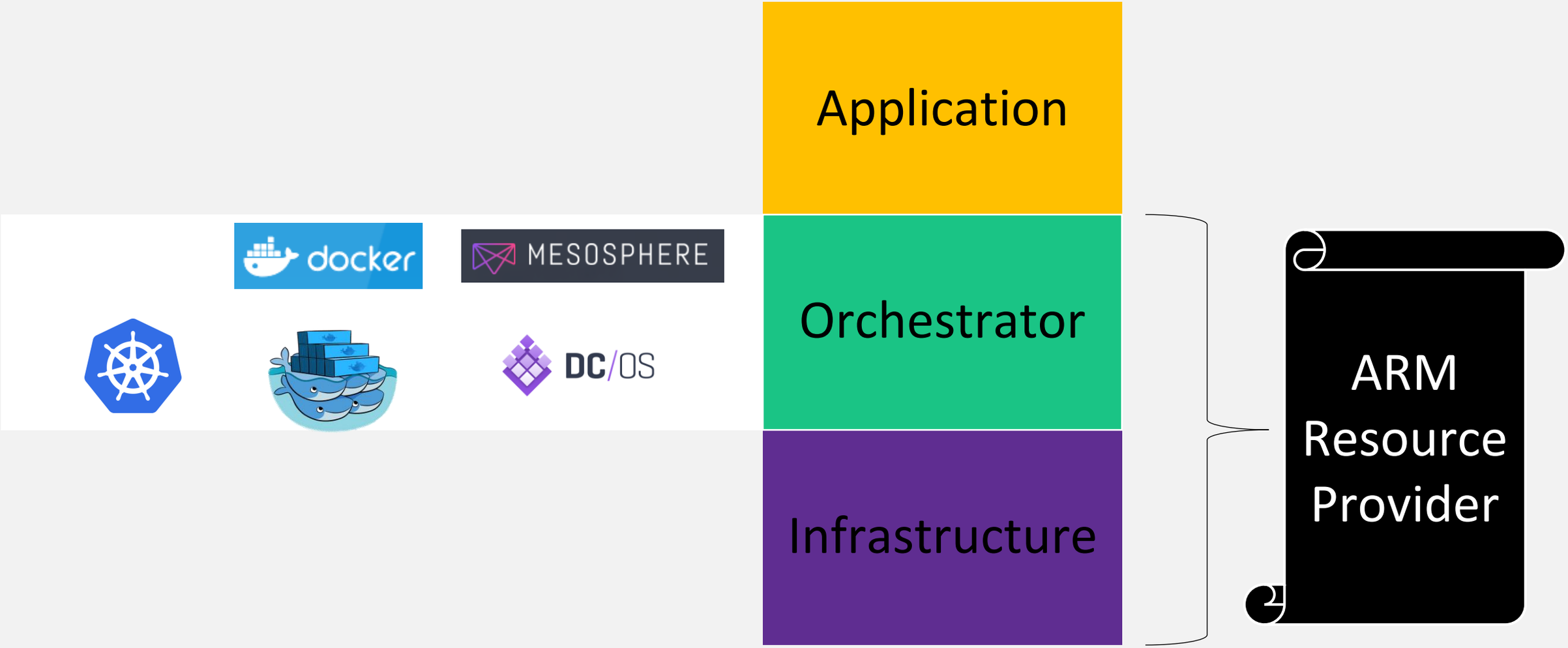
Azure Container Service (ACS)

- Create an optimized container hosting solution
- Standard tooling and API support
- Streamline provisioning of DC/OS, Docker Swarm, and Kubernetes (in preview)
- ACS engine is open source: https://github.com/Azure/acs-engine
- Leverage Azure Platform capabilities

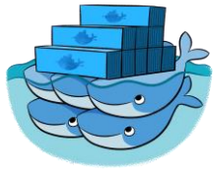
- Azure Resource Manager
- VM Scale Sets
- Networking
- Security



ACS and AKS on Azure



Azure Container Service (Cont.)



- Swarm provides native clustering for Docker
- Swarm serves the standard Docker API
- Docker daemon can use Swarm to transparently scale to multiple hosts on Azure Container Service

- Kubernetes is a container Orchestrator tool
- Automated Container Deployment
- Calling and management of containerized applications
- Windows Container support is in preview

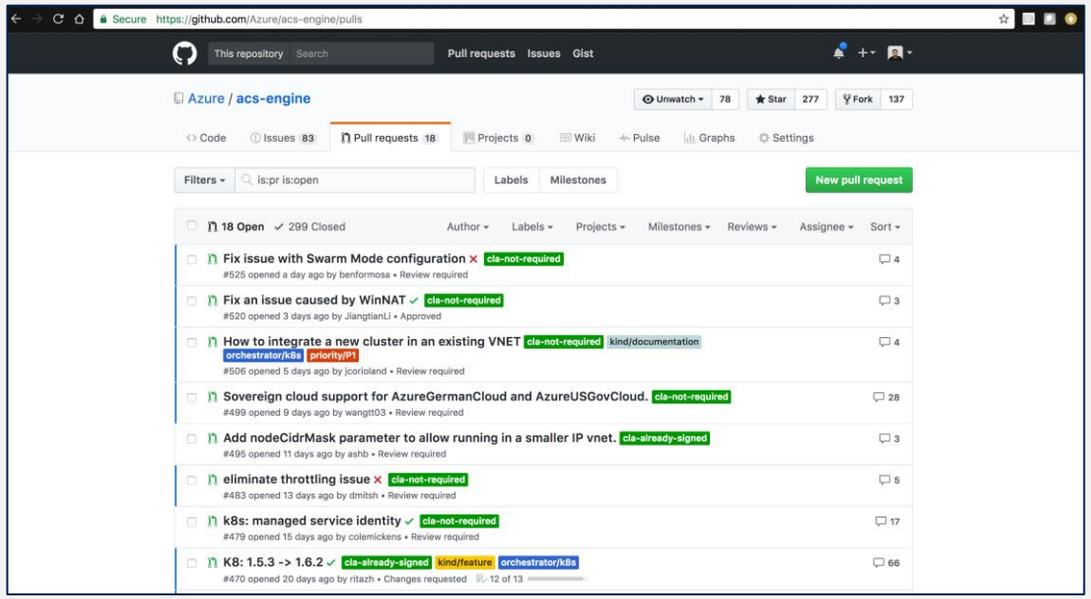
- DC/OS includes Marathon Orchestration platform for scheduling workloads
- Support for Docker-formatted containers
- REST APIs for communicating with Marathon

Azure Container Service Hybrid Story

- Azure Container Service on the Azure Cloud
  - Open Source Components
  - Non sticky
- Why open source only?
  - Take it and run it on premise
  - Use your own technical expertise
  - Use our partners Mesosphere and Docker
- Need Microsoft driven Hybrid?
  - Run in Azure Stack (coming)

Azure Container Service Engine

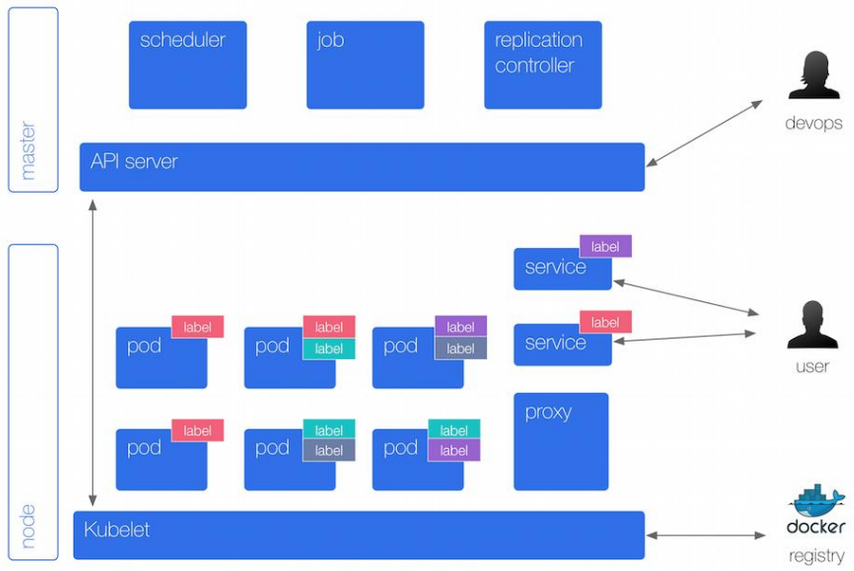
- Open source: http://github.com/Azure/acs-engine
- Customized deployment environments
- Existing VNET's
- Multiple agent pools
- Mixed Clusters (Windows and Linux)



Kubernetes

- Kubernetes is an open-source platform designed to automate deploying, scaling, and operating application containers.

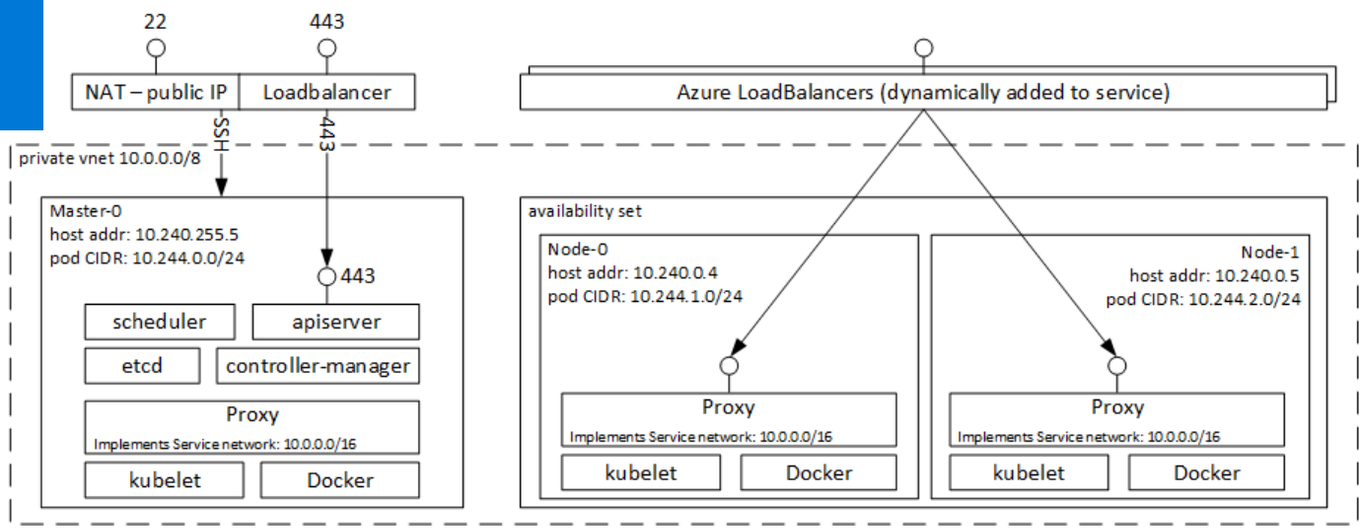
- With Kubernetes, you are able to quickly and efficiently respond to customer demand:



Kubernetes Components
Source: http://k8s.info/cs.html

- Deploy your applications quickly and predictably
- Scale your applications on the fly
- Roll out new features seamlessly
- Limit hardware usage to required resources only
- Supports Linux Container and Windows Containers (in preview)

More details: https://kubernetes.io/docs/concepts



Architectural diagram of Kubernetes deployed via Azure Container Service

Demonstration: *Kubernetes Cluster for Development*

Deploy a Kubernetes cluster on your laptop

Minikube

- Minikube is a tool that makes it easy to run Kubernetes locally.
- Minikube runs a single-node Kubernetes cluster inside a VM on your laptop for users looking to try out Kubernetes or develop with it day-to-day.
- Minikube supports Kubernetes features such as:

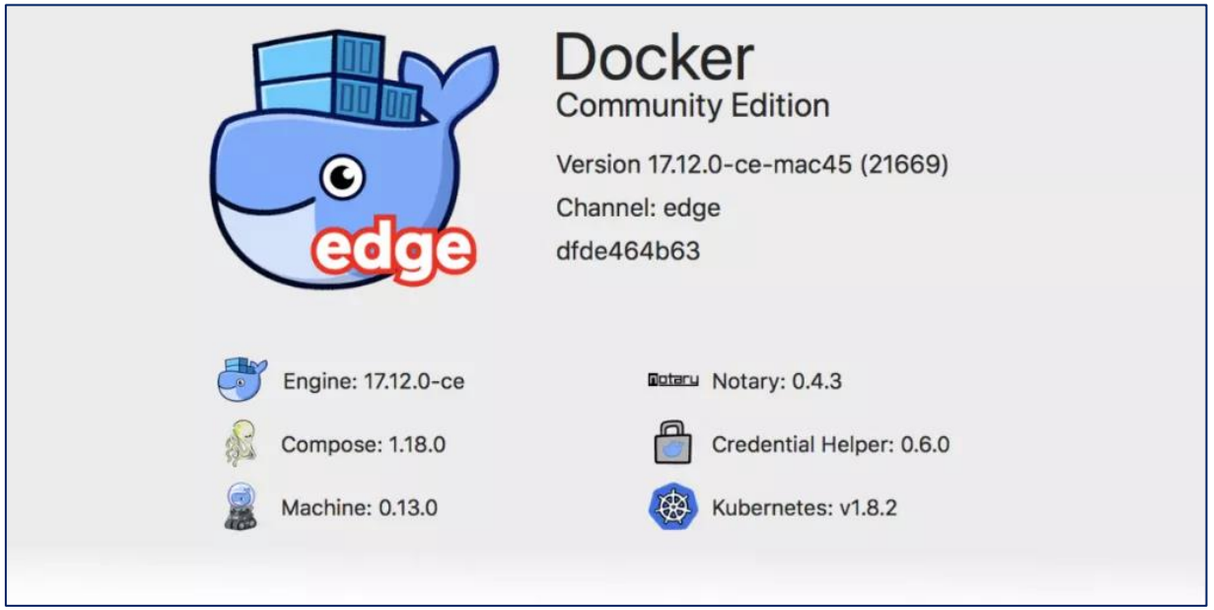
• DNS	• NodePorts	• ConfigMaps and Secrets	• Dashboards
• Container Runtime: Docker, rkt and CRI-O	• Enabling CNI (Container Network Interface)	• Ingress	

More details: **https://kubernetes.io/docs/getting-started-guides/minikube**

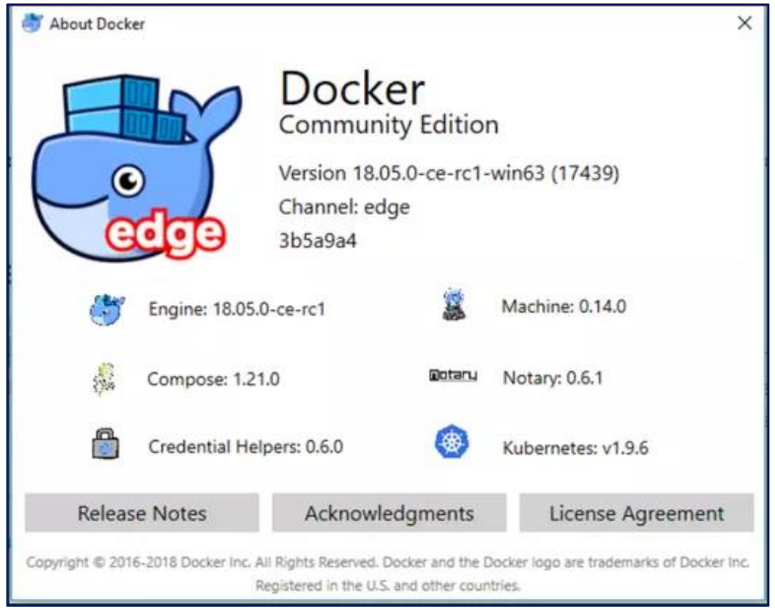
Docker + Kubernetes

- Since early 2018, Docker platform integrates with Kubernetes. This means that 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)



Docker for Mac



Docker for Windows

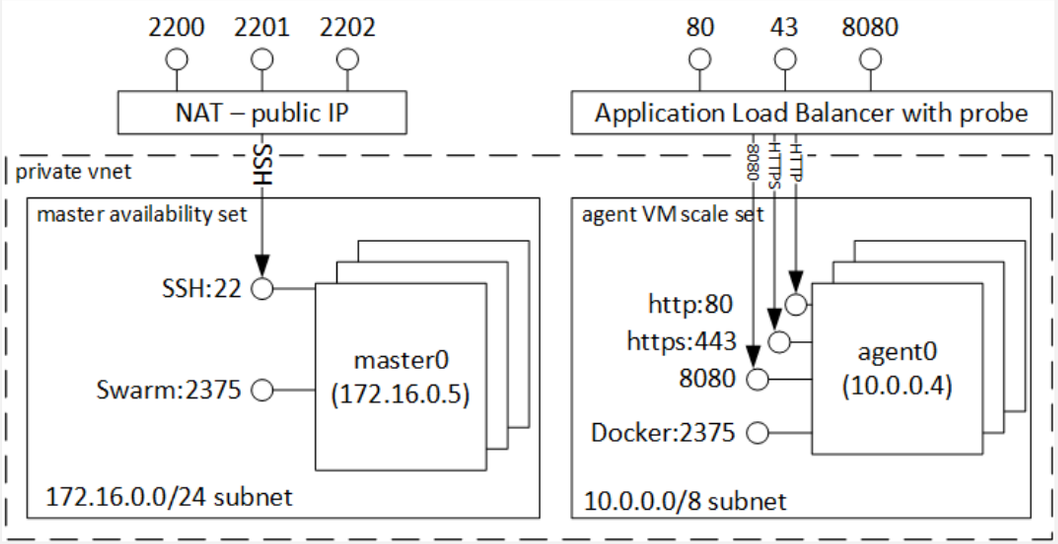
Demonstration: *Working with Kubernetes Minikube*

Minikube Overview

Docker Swarm

- Docker Swarm provides an environment for deploying containerized workloads across a pooled set of Docker hosts
- Docker Swarm is native clustering for Docker
- Docker Swarm uses the native Docker API. The workflow for managing containers on a Docker Swarm is almost identical to what it would be on a single container host.

NOTE: The Docker Swarm orchestrator in Azure Container Service uses legacy standalone Swarm. Currently, the integrated Swarm mode (in Docker 1.12 and higher) is not a supported orchestrator in Azure Container Service.



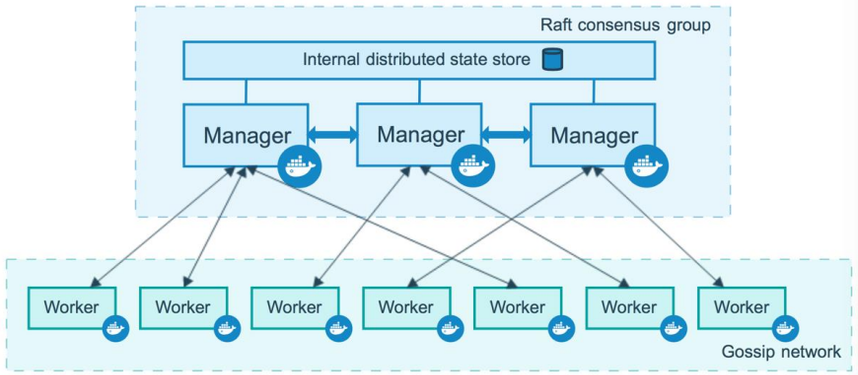
Architectural diagram of Docker Swarm deployed via Azure Container Service

Docker Swarm Mode

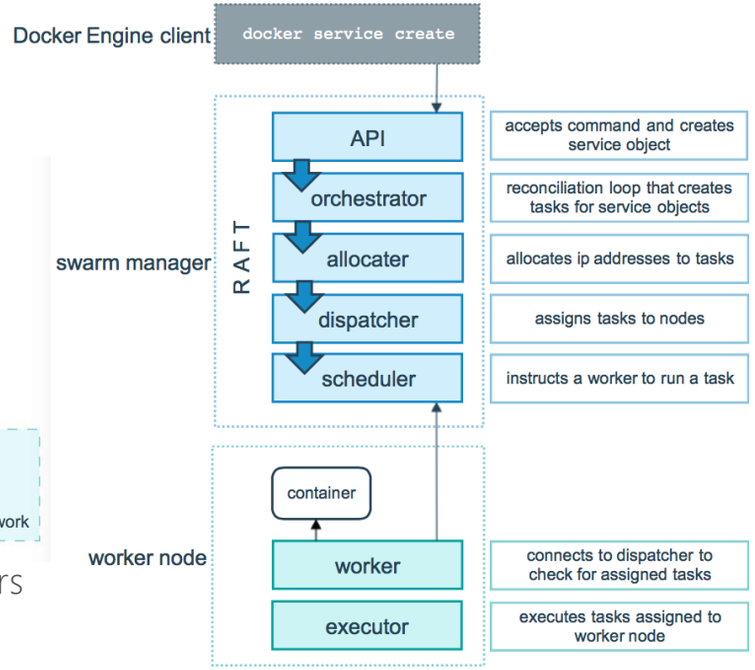
Swarm mode is a Docker feature that provides built in container orchestration capabilities, including native clustering of Docker hosts and scheduling of container workloads.

Only available with Docker Engine 1.12 or above.

Swarm supported on Windows but have few limitations: https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-containers/swarm-mode#limitations

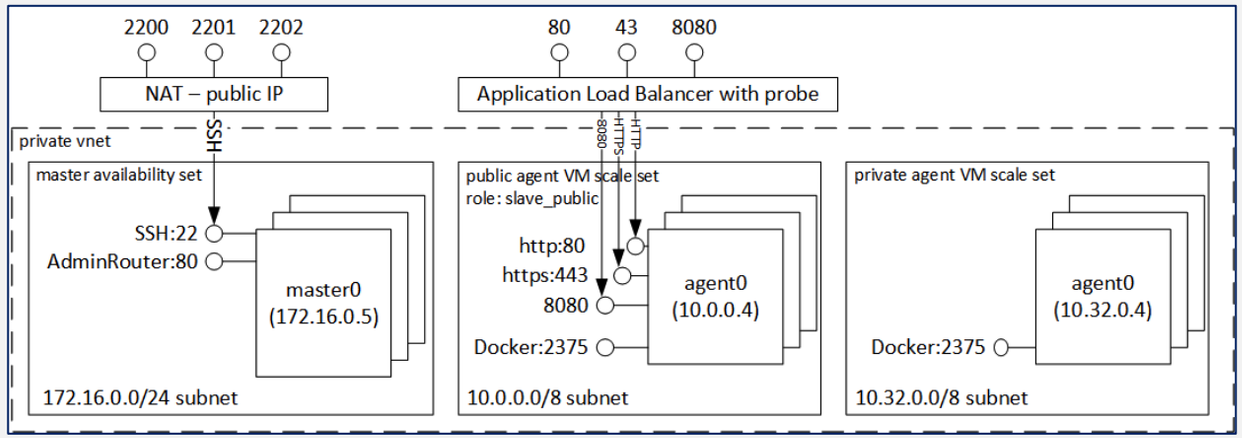


Swarm Mode Nodes: Managers and Workers

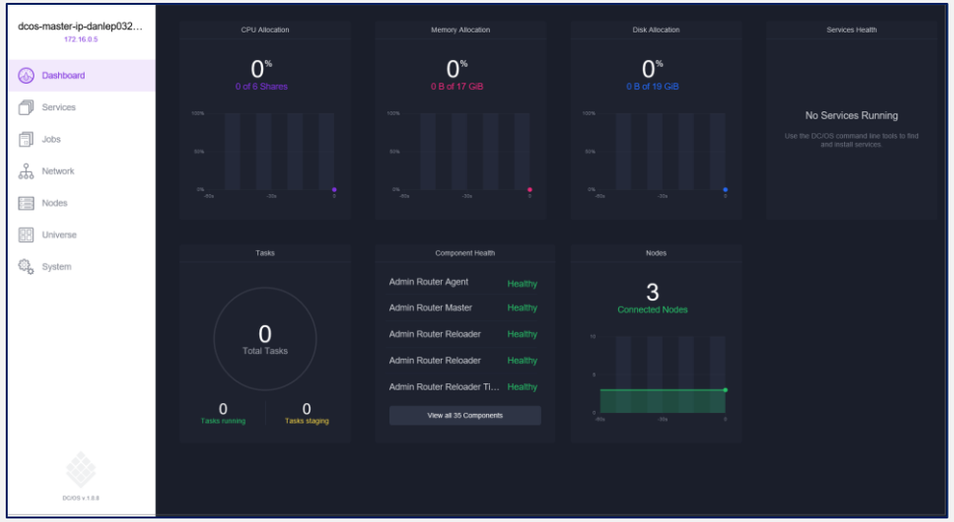


Swarm Mode: Service create requests and scheduling of tasks to worker nodes

DC/OS



Architectural diagram of DC/OS deployed via Azure Container Service



Marathon UI

DC/OS is a distributed operating system – powered by Apache Mesos – that treats collections of CPUs, RAM, networking and so on as a distributed kernel.

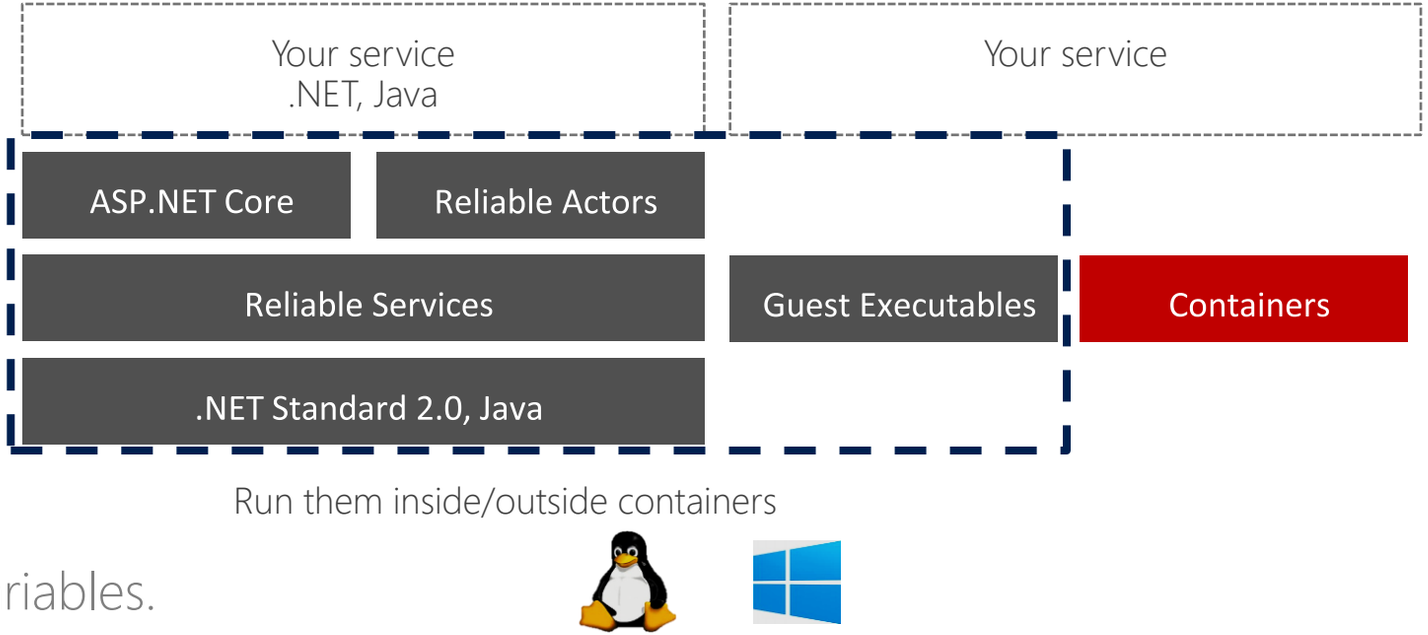
- Mix of container and non-containerized apps
- Big data and analytics workloads
- Enterprise scale and support

Marathon: Framework sits on top of DC/OS that manages scheduling and executing compute workloads

Azure Service Fabric

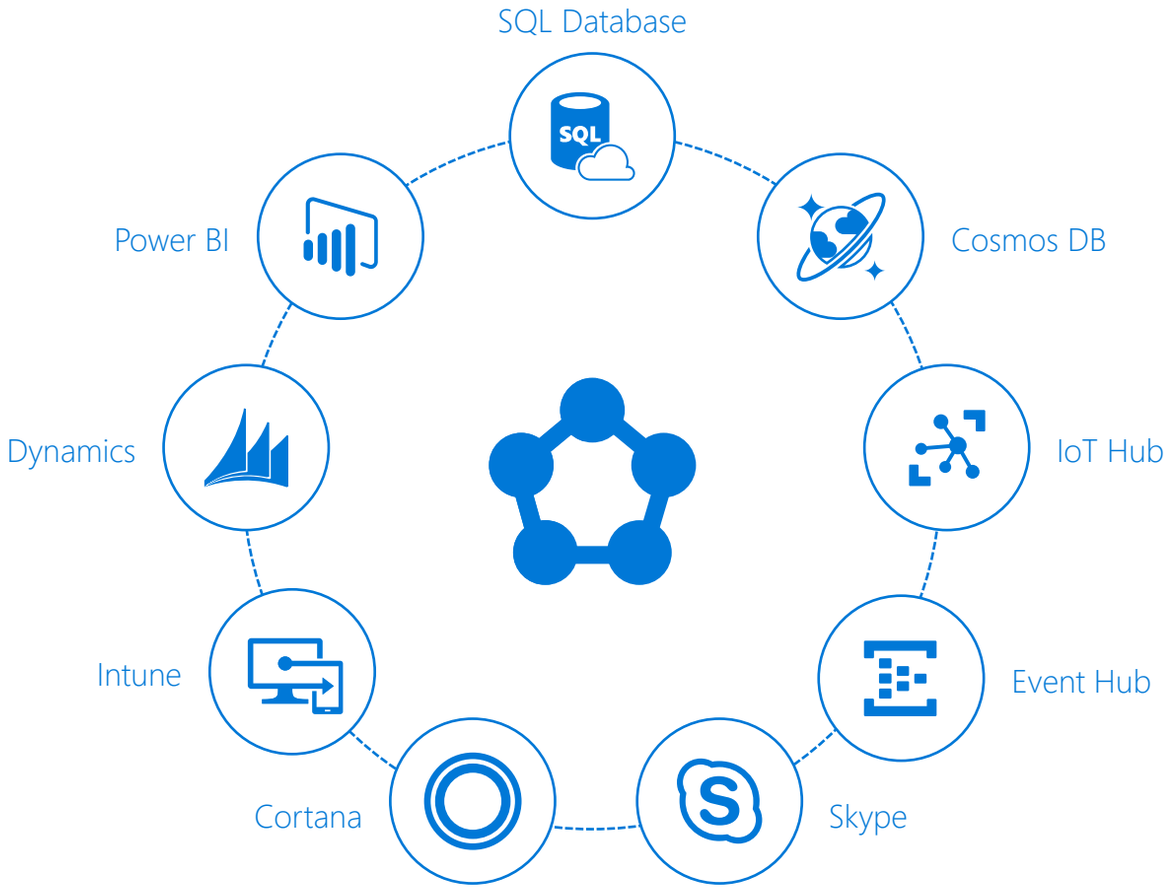
Azure Service Fabric is an orchestrator of services across a cluster of machines, with years of usage and optimization in massive scale services at Microsoft.

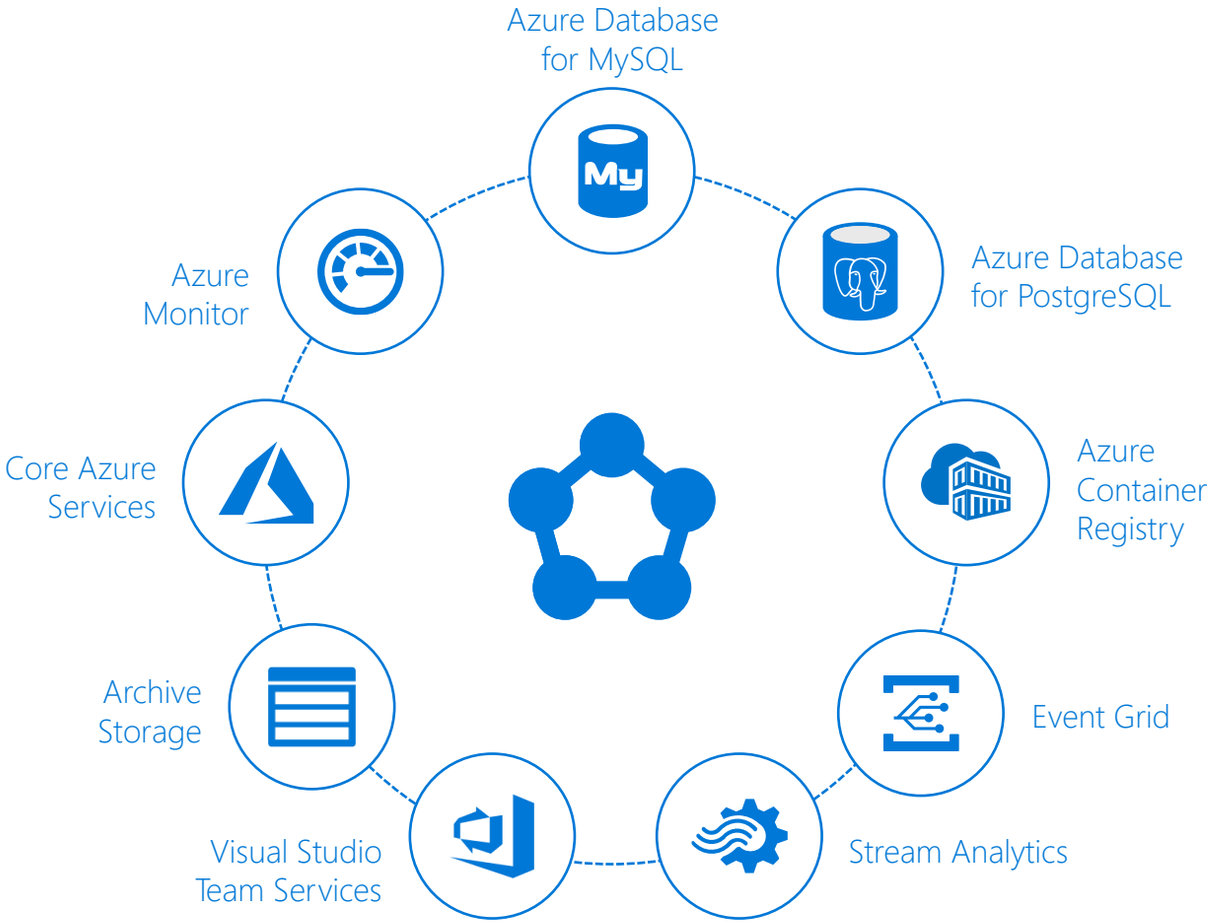
- Supports containers on both Linux and Windows.
- Container image deployment and activation.
- Container Resource governance.
- Repository authentication.
- Container port to host port mapping.
- Container-to-container discovery and communication.
- Ability to configure and set environment variables.



Service Fabric Containers Roadmap:
https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-containers-overview#containers-and-service-fabric-roadmap

Example PaaS Services Built On Service Fabric





Service Fabric vs Kubernetes

	Service Fabric	Kubernetes
Components	Orchestrator + Programming Model (Reliable Services/Actors)	Orchestrator
Portability	Run in Azure or standalone (on-premises or any Cloud provider)	Run in Azure, on-premises or any Cloud Provider
Open Source	https://github.com/Microsoft/service-fabric	https://github.com/kubernetes/kubernetes
Implementation	Driven by Microsoft. More opinionated.	Community Driven. Less Opinionated.
Support	Microsoft	Community
Modularity	All Integrated Environment but opened to some third-party integration (Splunk, Containers, SDK, …)	Pluggable Environment where components need to be chosen (SDK, monitoring, persistence, network, developer tools, …)
Stateful Workload	More mature (data replicated in RAM/SSD disk across nodes)	Less mature (rely on shared volumes)
Deployment Model	Rolling Upgrade by default, Blue Green deployment doable but involved	Rolling Upgrade by default, Blue Green deployment intuitive with Services
Windows	More mature	Less mature: ACS & ACS-engine not GA for Windows
Linux	Less mature: no standalone clusters yet	More mature

Demonstration: *Azure Kubernetes Service*

Deploy application container into AKS

Azure Container Instances

Azure Container Instances is great solution 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

Azure Container Instances is NOT a great solution for:

- Full container orchestration
- Service discovery across multiple containers
- Automatic scaling
- Coordinated application upgrades

For all of above Microsoft recommend the Azure Container Service.

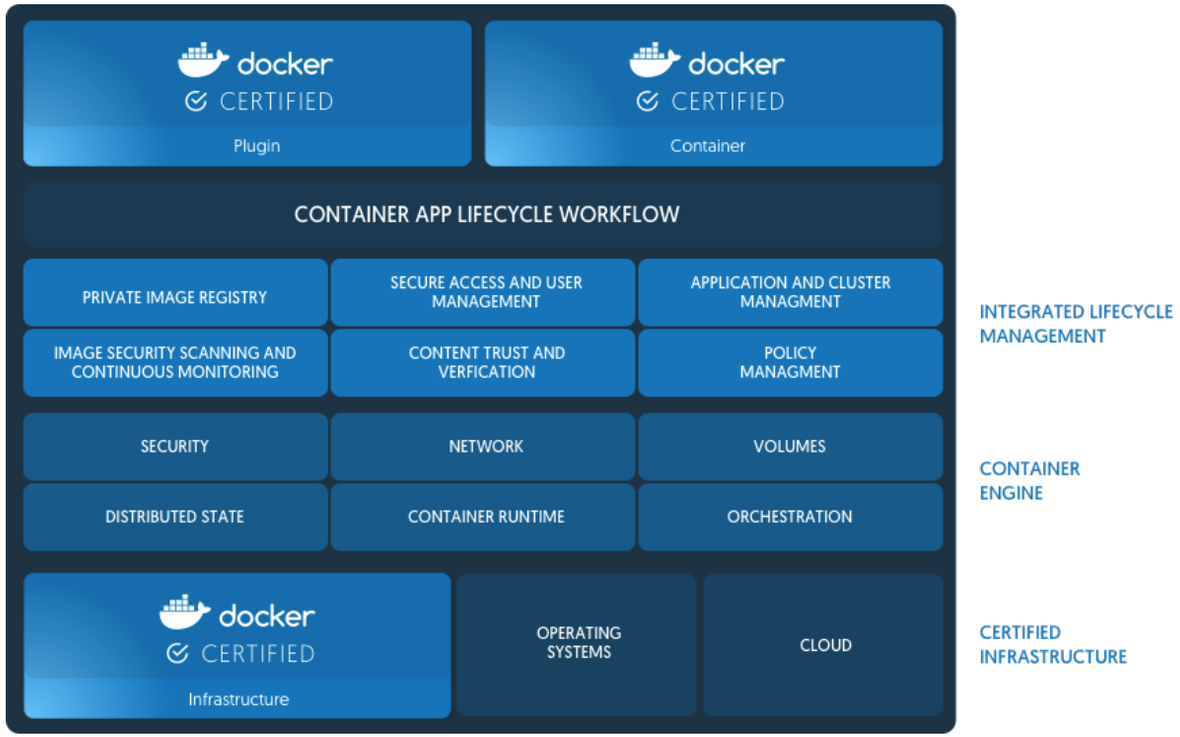
Docker Enterprise Edition for Azure

Docker Enterprise Edition for Azure (Standard/Advanced) is a one-click deploy of highly scalable Docker Datacenter based on Docker and Azure best practices.

Features:

- Docker Universal Control Plane (UCP) is an enterprise grade cluster management solution to deploy and manage containerized application.

- Docker Trusted Registry (DTR) is a containerized application that allows you to store and manage your Docker images securely inside your firewall while integrating to your CI workflow.



- Available through Azure Market: https://azuremarketplace.microsoft.com/en-us/marketplace/apps/docker.dockerdatacenter
- Details: https://www.docker.com/enterprise-edition