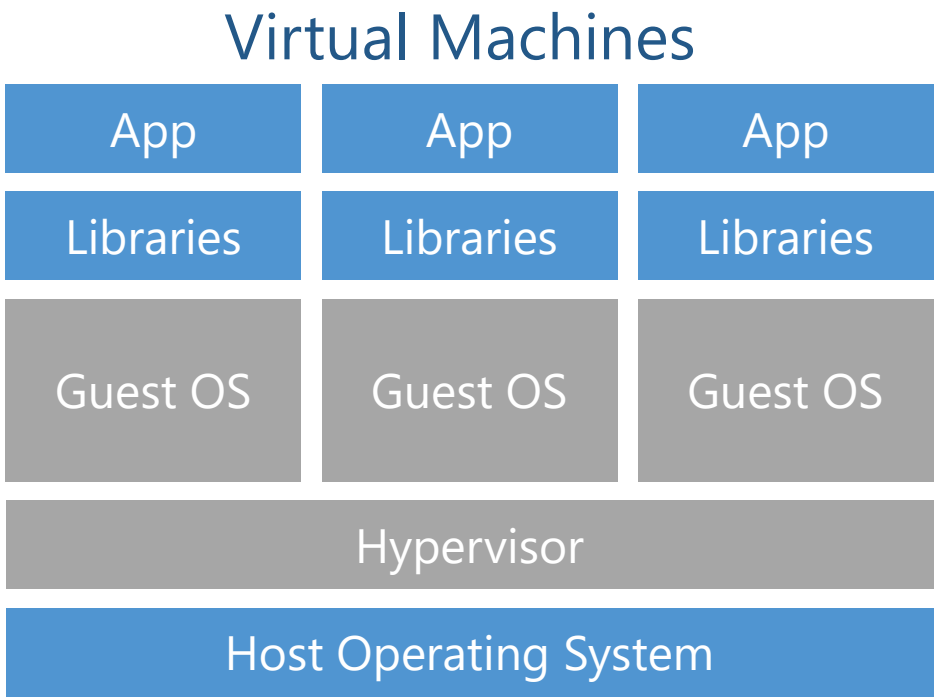
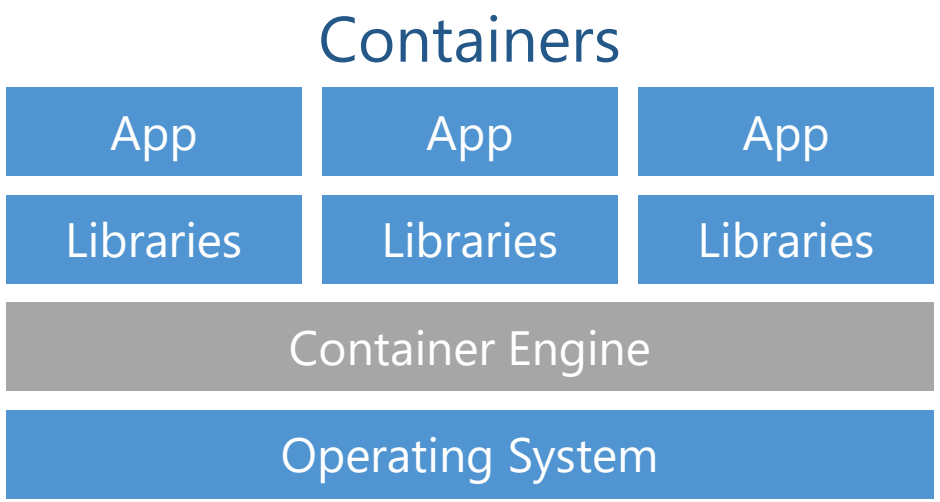
Container Services

Containers

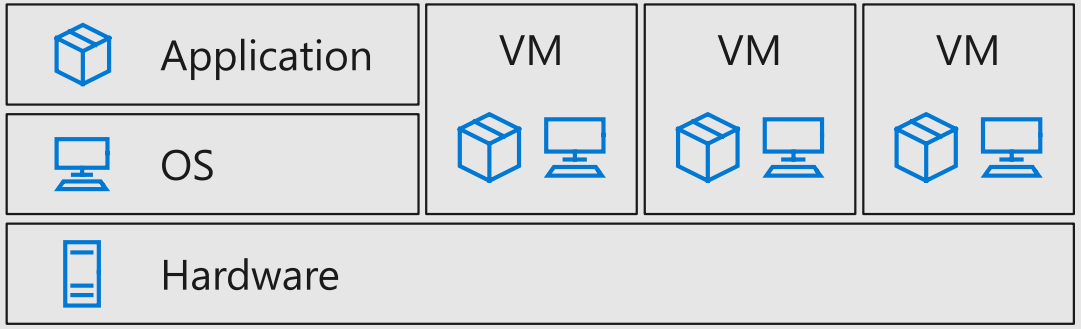
- Lightweight alternative to virtual machines
- Smaller, less expensive, faster to start up, and self-contained



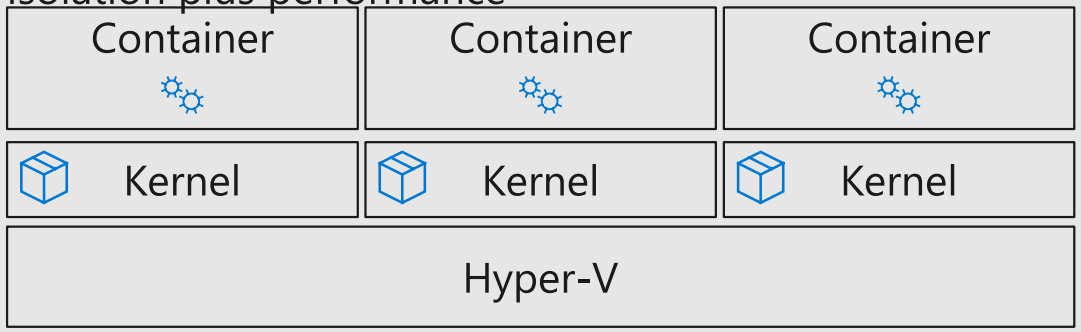


What is a container?

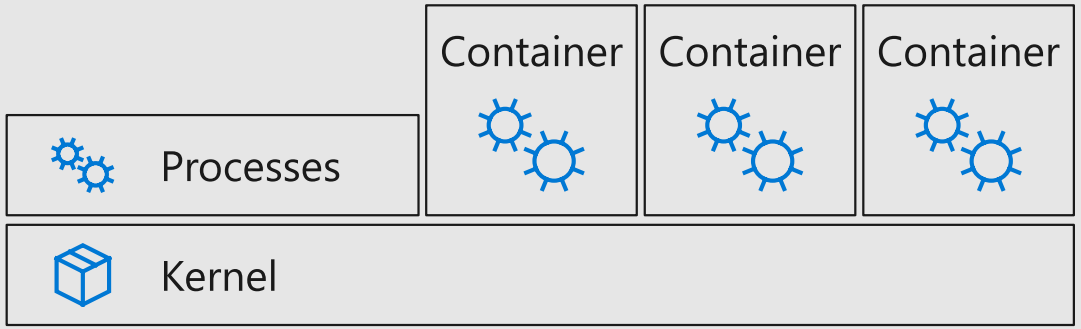
Traditional virtual machines = hardware virtualization



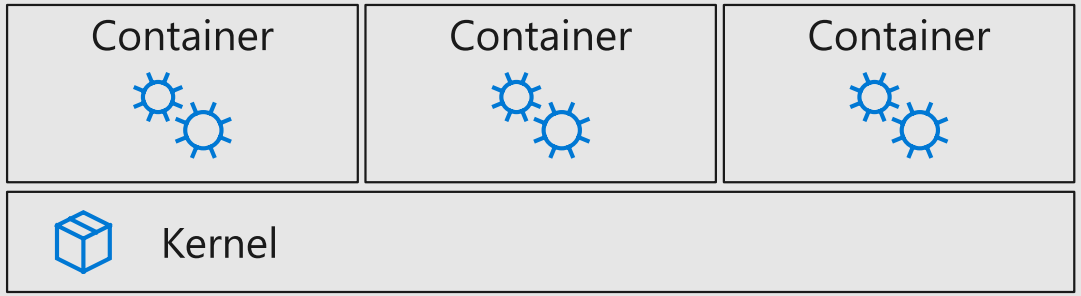
Hyper-V containers—
Isolation plus performance



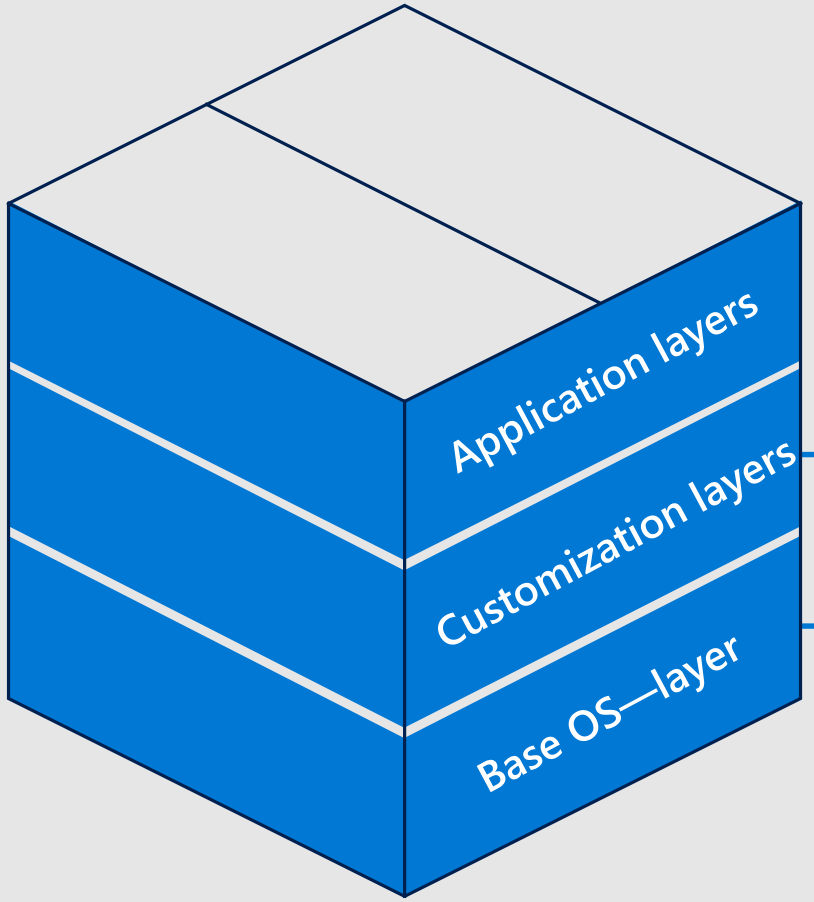
Containers = operating system virtualization



Windows server containers—
Maximum speed and density



Anatomy of a Windows container



Application code

Pre-requisites | Customizations

App Service supports the long-term servicing channel of Windows Server 2016

i.e., Windows Server core 2016 LTSC | i.e., Windows Server Nano 2016 LTSC

Windows container best practices

Choose base image carefully

- Core/Nano—LTSC/SAC
- Choose cached images in order to benefit from speed of pull

Layers

- Minimize image layers

Dockerfile optimizations—https://aka.ms/dockerfileoptimization

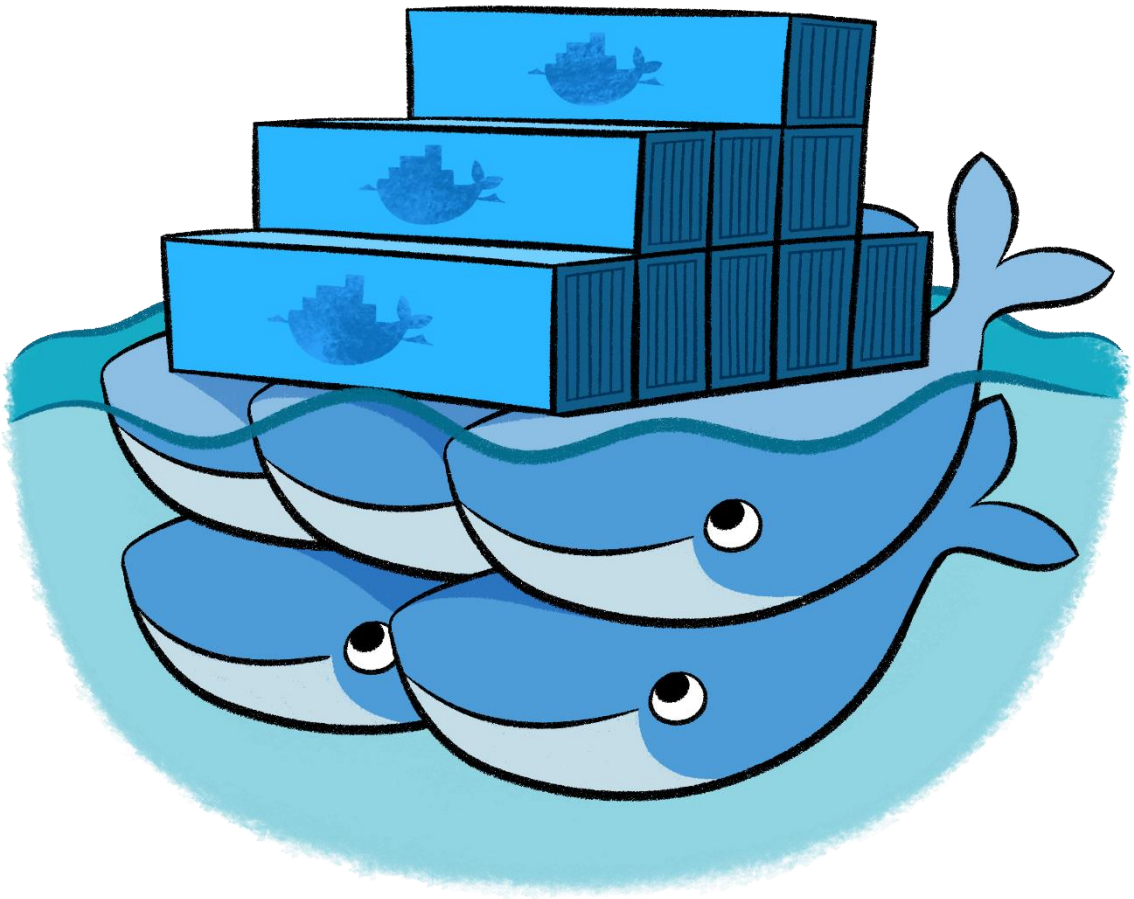
- Image size
  - Group related actions
  - Remove excess files
- Build speed
  - Multiple lines
  - Ordering of instructions
- Cosmetic optimizations

Docker

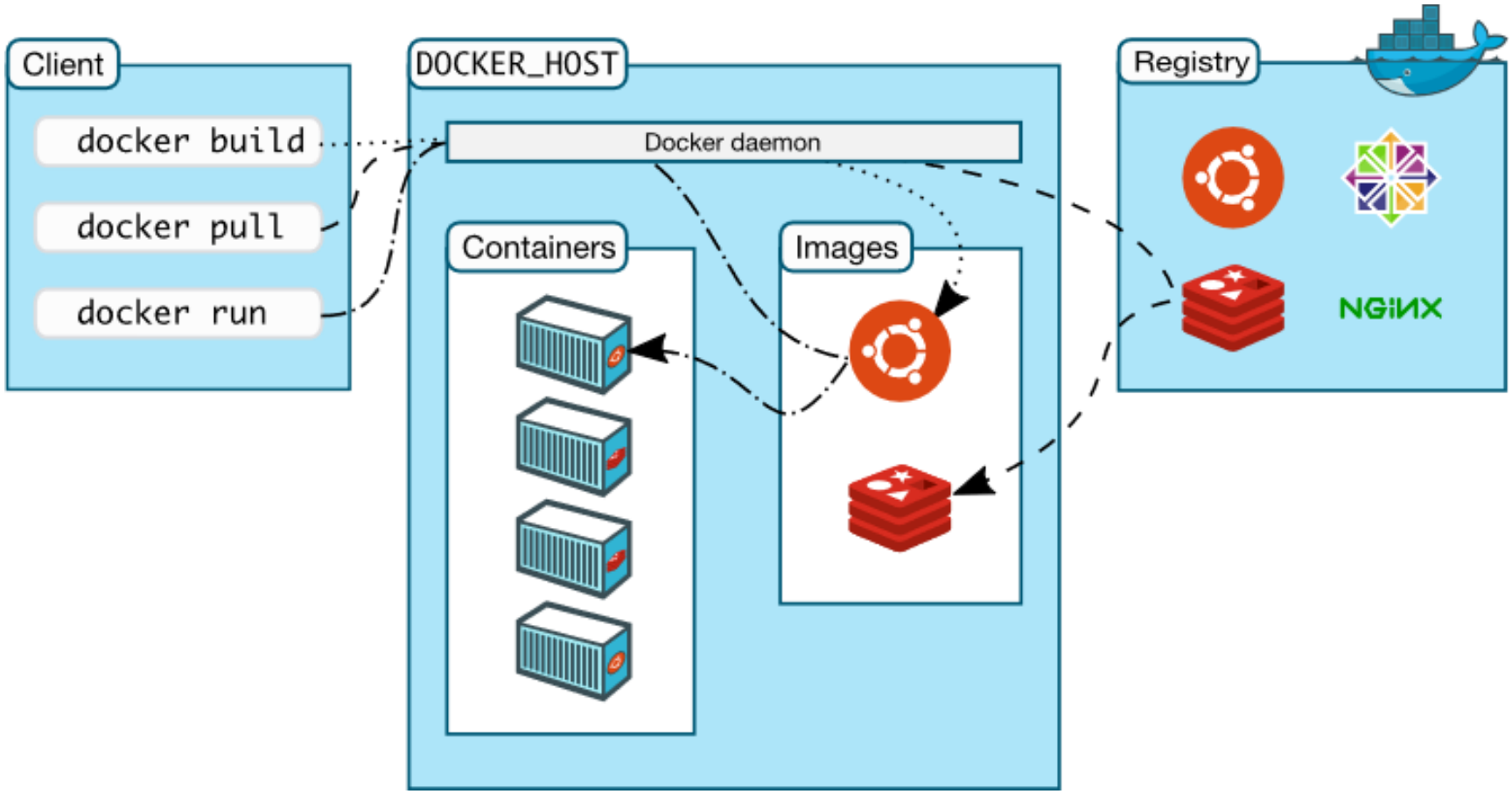
- Leading open-source containerization platform

  Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in

- Supported natively in Azure

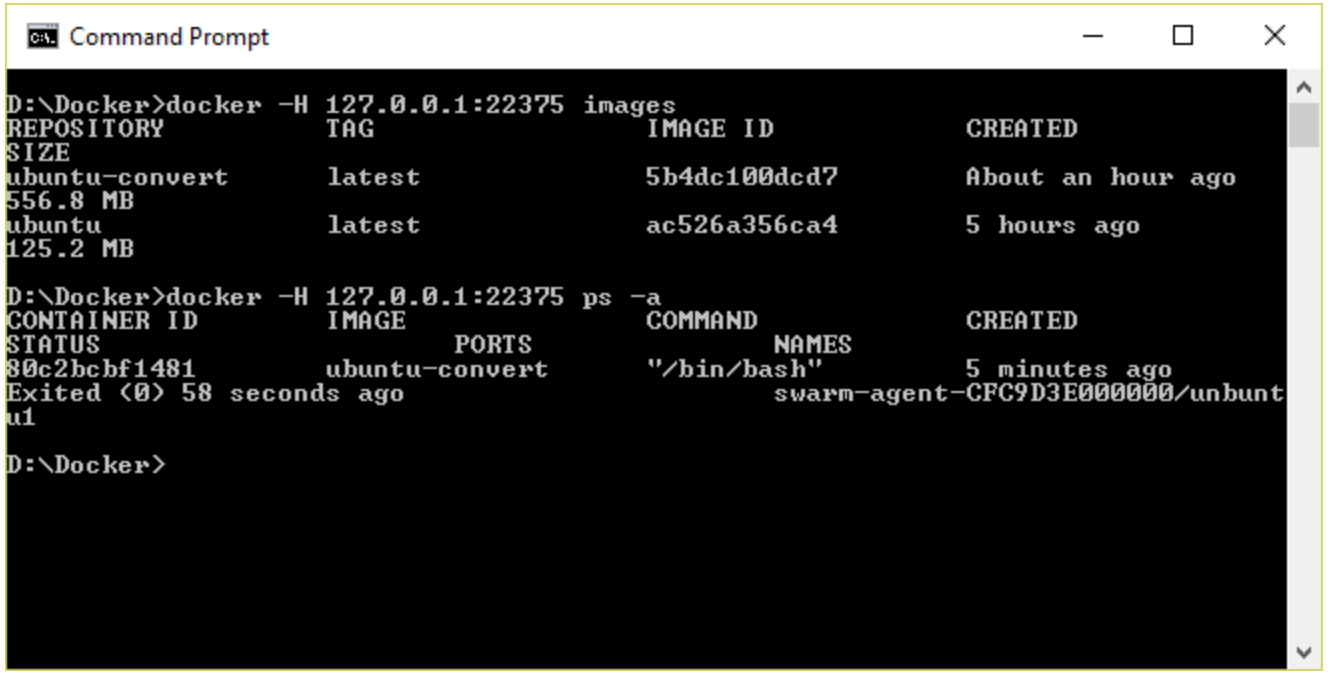


Docker Architecture



Docker CLI

- Command-line interface for Docker, available for Linux, OS X, and Windows (available separately or as part of Docker Toolbox)



Running a Container

```
docker run -i -t ubuntu /bin/bash
```

Docker CLI command

Run container with interactive terminal

Pull "ubuntu" image from Docker Hub or local registry

Command to execute in the container

Common Docker CLI Commands

`docker run` - Use an image to run a container

`docker pull` - Pull an image from a registry

`docker build` - Build a Docker image

`docker images` - List available Docker images

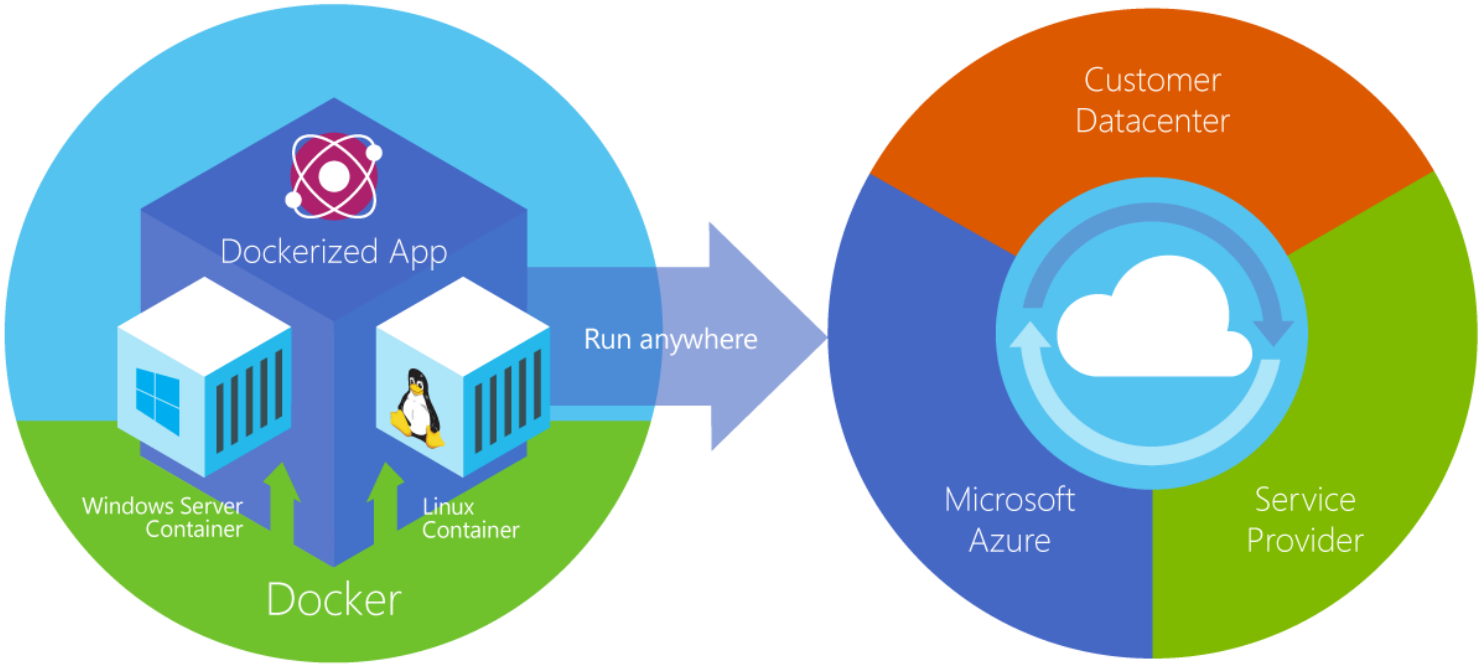
`docker ps` - List running Docker containers

`docker exec` - Execute a command in a container

`docker stop` - Stop a running container

Azure Container Service

- Provides robust, ready-to-use Docker hosting environment
- Uses open-source orchestration tools (DC/OS and Swarm)

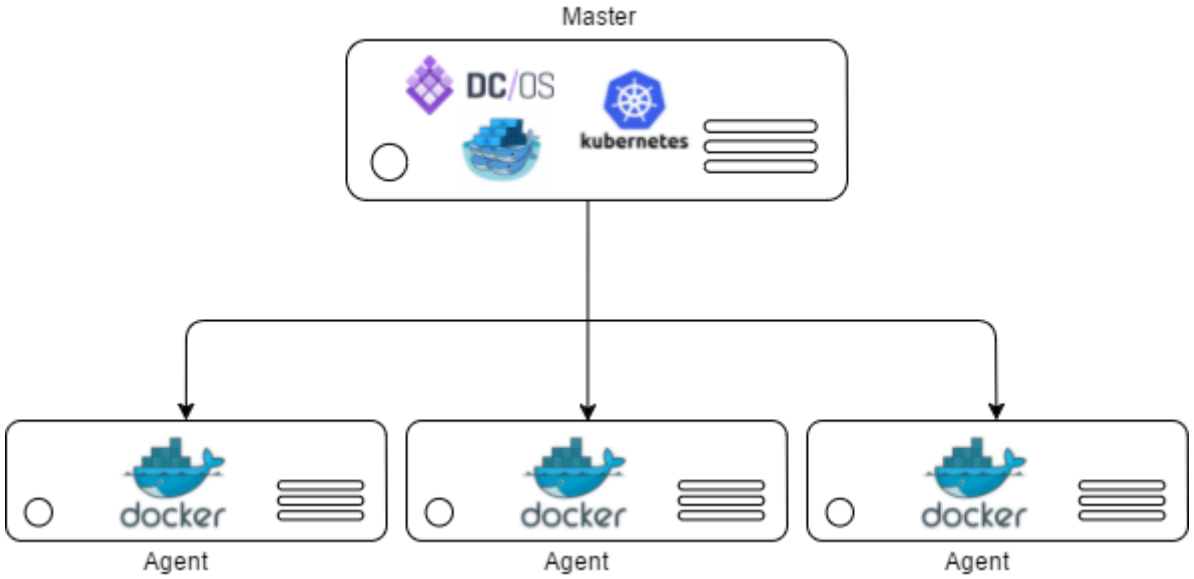


Container Orchestration

- Facilitates deployment and management of containers
- Containers by design are intended to be deployed in large volumes with some applications using dozens to even thousands of containers
- With this type of scale, automating container deployment and management with orchestration software becomes necessary
- Azure Container service supports Kubernetes, DC/OS, and Docker Swarm

Container Clusters

- Facilitate load balancing, scalability, and high availability
- A cluster is composed of master nodes which control the orchestration, and agent nodes that host the containers



Kubernetes

- Open-source orchestration engine from Google
- Provides a robust framework for container orchestration, yet remains lightweight and scalable
- Supported by Azure Container Service and tightly integrated with ACS, allowing Kubernetes to modify deployments

kubernetes
by Google

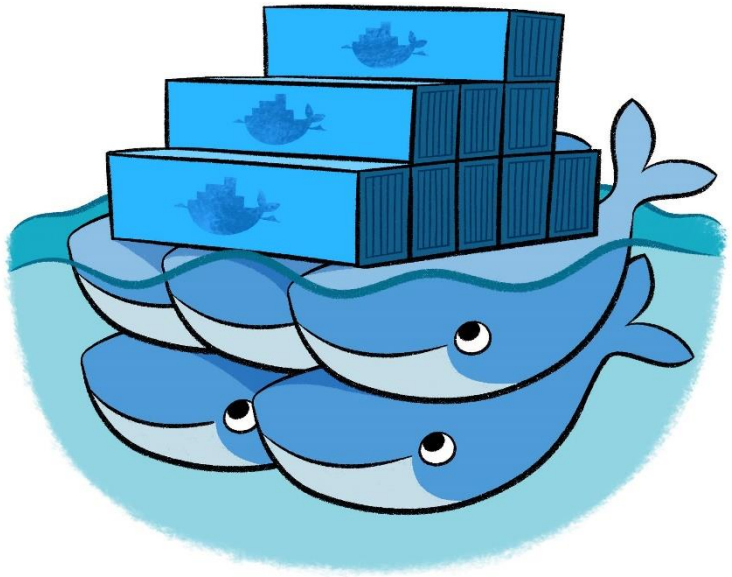
DC/OS

- Datacenter Operating System built on Apache Mesos
- Creates logical data centers and abstracts underlying hardware
- Provides resources traditionally provided by infrastructure, including networking, DNS, and load balancing
- Natively supported by Azure Container Service

DC/OS

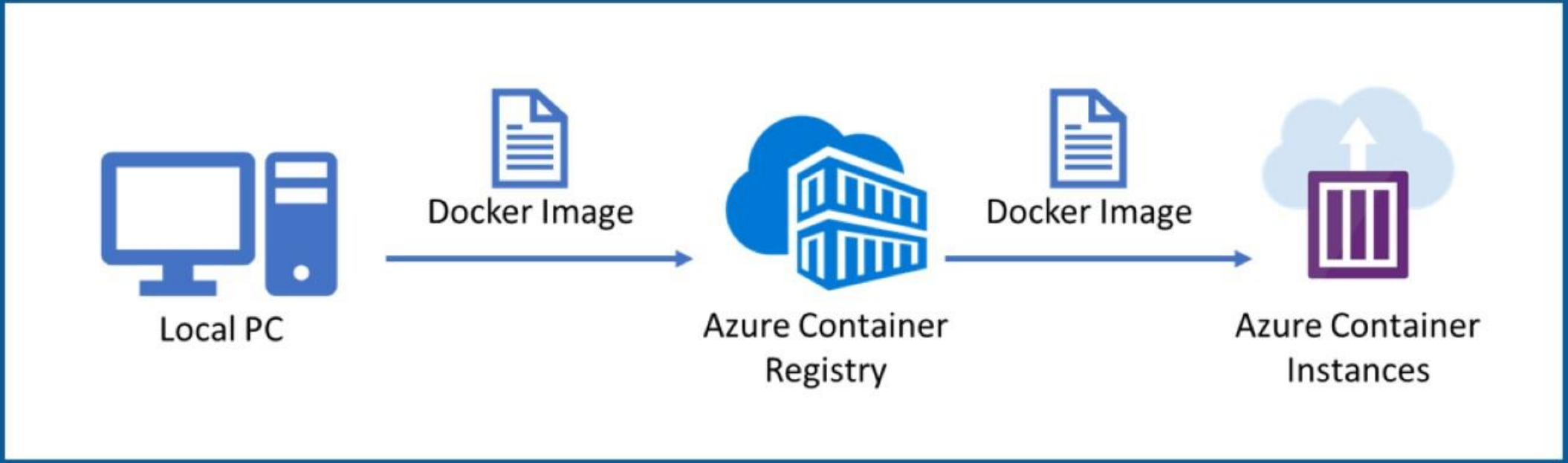
Docker Swarm

- Docker’s own orchestration engine
- Current releases of the Docker engine have “Swarm Mode” built in and can many of the same things that other orchestration engines do
- Lacks a GUI, but makes up for it with tight integration with Docker
- Natively supported by Azure Container Service



Azure Container Registry

Introduction



Key Concepts

Registry	Repository
Image	Container

SKUs

Basic

Standard

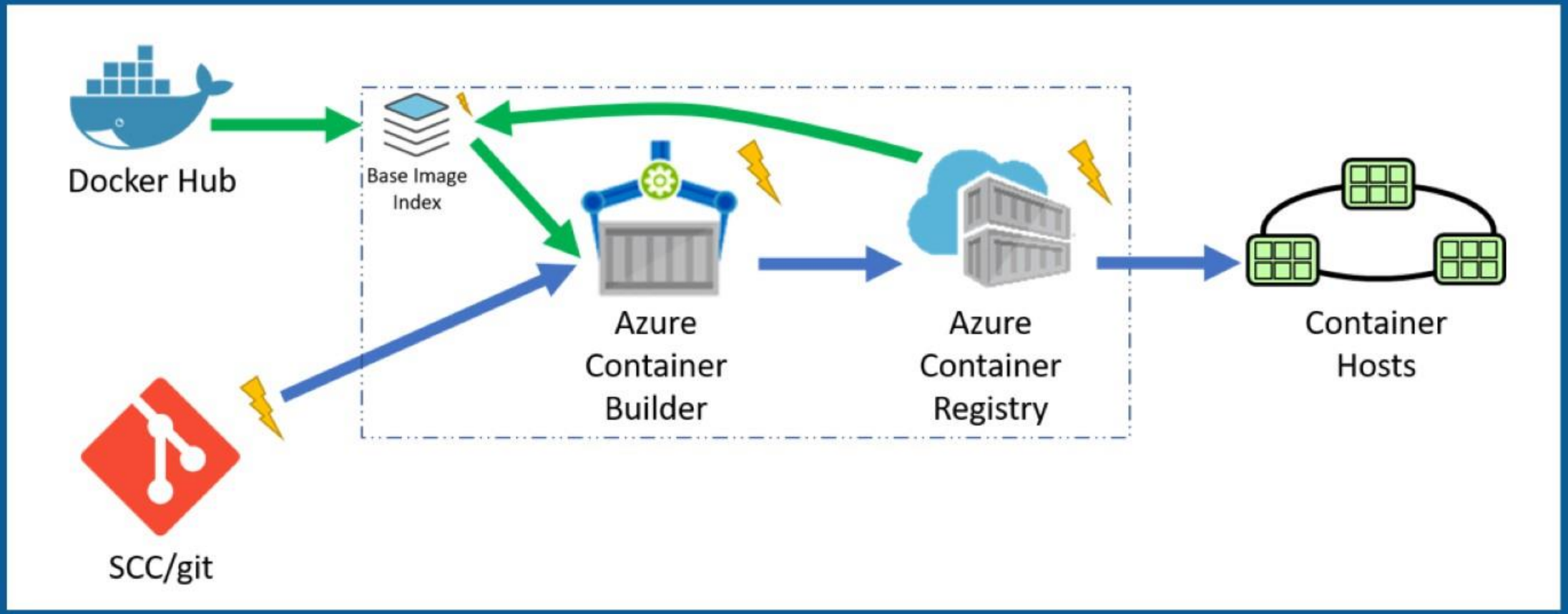
Premium

https://docs.microsoft.com/en-us/azure/container-registry/container-registry-skus

Image Storage

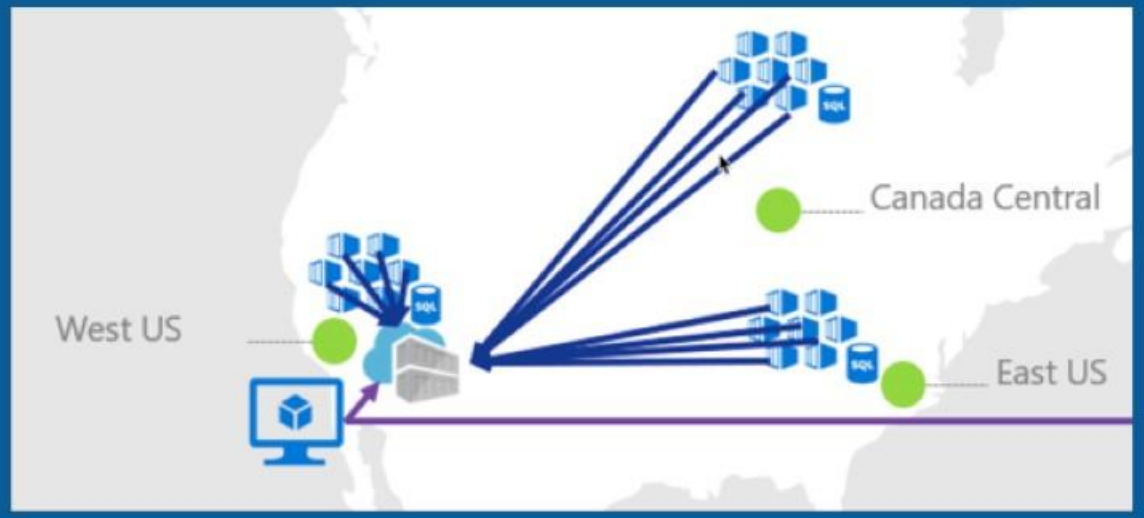
- All container images are encrypted at rest
- Encryption-at-rest for image data security
- Geo-redundancy for image data protection

Azure Container Registry Build Tasks



Geo-Replication

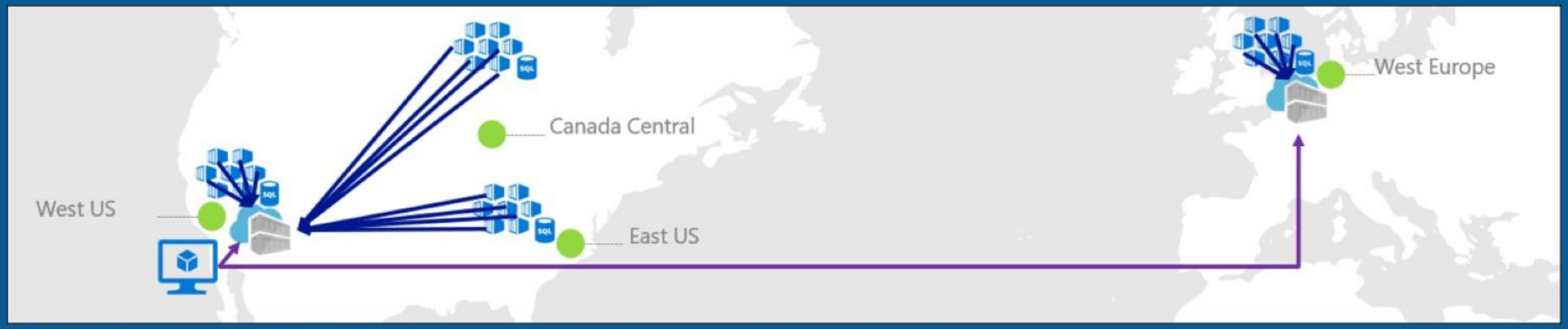
- Single registry / image / tag names
- Network-close registry access
- No additional egress fees
- Single management of registry



Geo-Replication Example Use Case

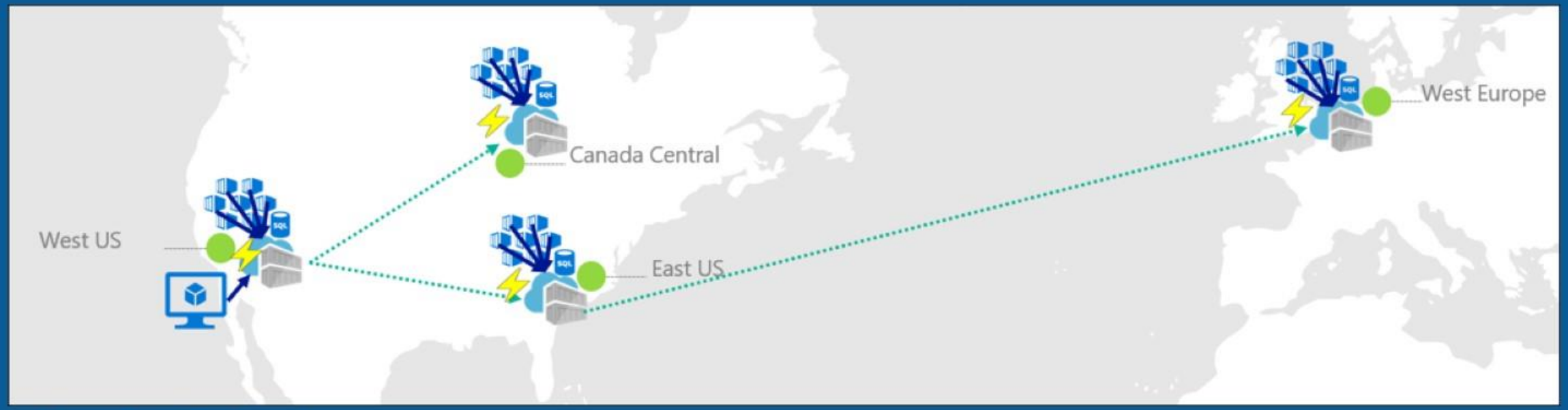
Pushing to multiple registries

Geo-Replication Example Use Case



Pulling from multiple registries

Geo-Replication Example Use Case



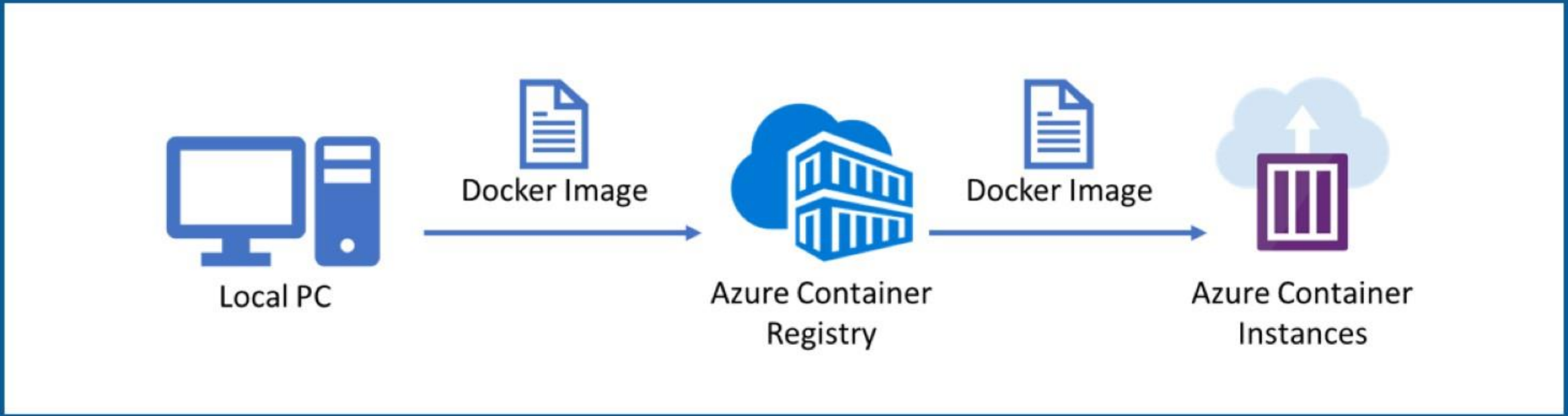
Using Geo-Replication

ACR Best Practices

- Network-close deployment
- Geo-replicate multi-region deployments
- Repository namespaces
- Dedicated resource group
- Manage registry size

Azure Container Instances

Introduction



Public IP & DNS name

Hypervisor-level security

Custom sizes

Persistent storage

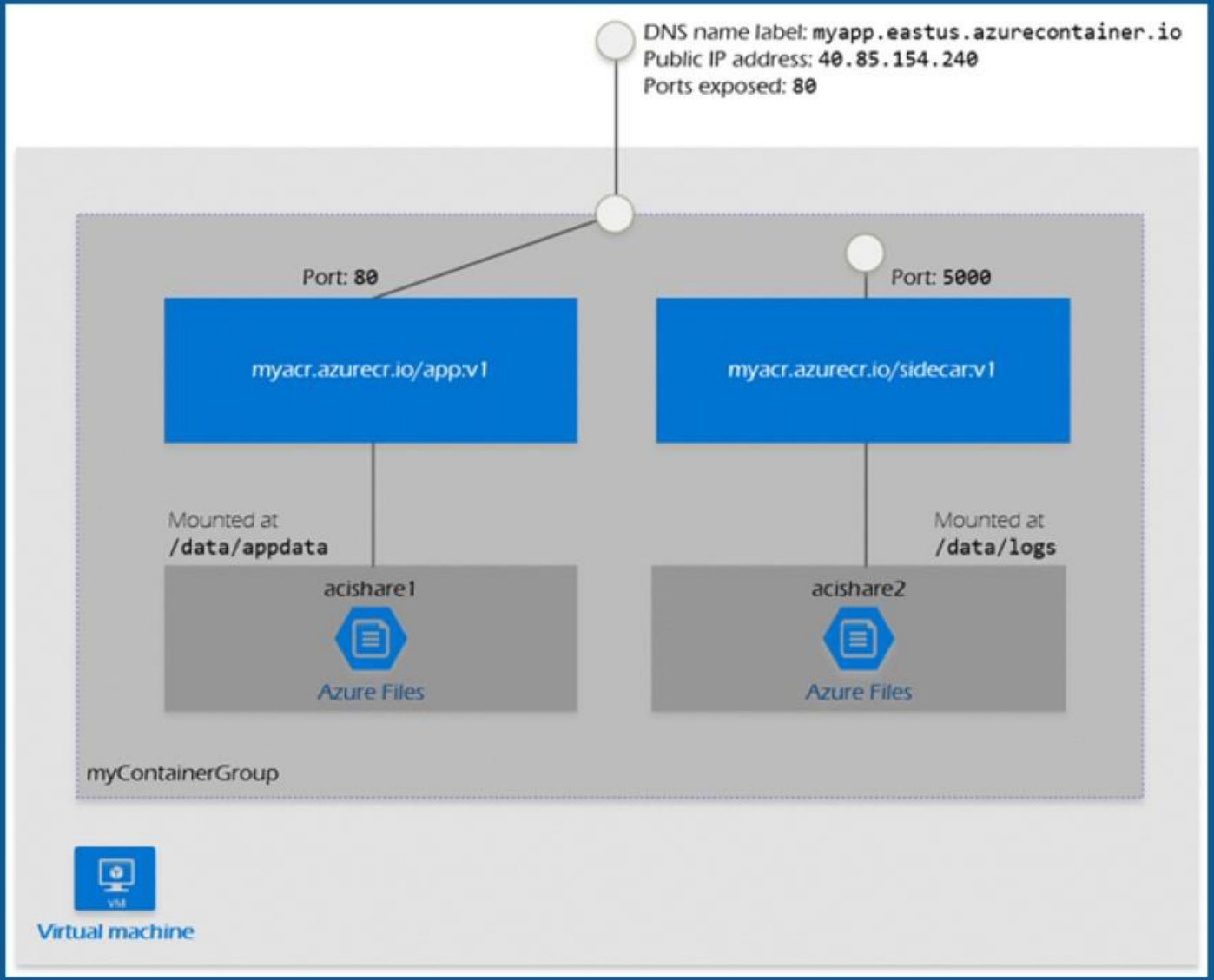
Co-scheduled groups

Container Orchestrators

- Scheduling
- Affinity / Anti-affinity
- Health monitoring
- Failover
- Scaling
- Networking
- Service discovery
- Coordinated application upgrades

DC/OS

Container Groups



Container Groups

- Deployment
  - Minimum resource allocation of 1 vCPU and 1 GB of memory
  - Containers can be provisioned with less than 1 vCPU and 1 GB of memory
- Networking
  - Share an IP address and a port namespace
  - Expose the port on the IP address to enable external clients
- Storage
  - Mount external volumes
  - Map volumes to specific paths

Containerized Tasks (Restart Policy)

- Always
  - Containers in the container group are always restarted
  - This is the default when no restart policy is specified at container creation
- Never
  - Containers in the container group are never restarted
  - The containers are run at most once
- OnFailure
  - Containers in the container group are restarted only when the process executed in the container fails
  - The containers are run at least once

Kubernetes vs Docker

Comparison Chart

Kubernetes	Docker
Kubernetes is an ecosystem for managing a cluster of Docker containers known as Pods.	Docker is a container platform for building, configuring and distributing Docker containers.
Kubernetes is not a complete solution and uses custom plugins to extend its functionality.	Docker uses its very own native clustering solution for Docker containers called Docker Swarm.
Load balancing comes out of the box in Kubernetes because of its architecture and it's very convenient	The load balancer is deployed on its own single node swarm when pods in the container are defined as service.
Takes relatively more time for installation.	Setup is quick and easy compared to Kubernetes.

	KUBERNETES	MESOS / DCOS	Docker Swarm
Years released	2+ (with 15 years experience Borg)	8+	2+
Community	One of the top projects on GitHub (ranked nr. 4)	Mesosphere is the top contributor	Most Swarm contribs come from Docker Inc.
Vendor/Community-Maintained Offerings including PaaS	70+	Mesosphere	Docker / Moby
Public Cloud Service Providers	Google, Azure, AWS, OTC (Open Telekom Cloud, OpenStack based) (10+ AWS community supported versions)	Azure AWS Mesosphere	Azure
Strengths	Clear Market Leader: Largest adoption and interest amongst developers and the enterprise True Federation capabilities	Used by a few large organizations at massive scale (e.g. Twitter)	Mostly controlled by a single vendor who can decide product direction
Weakness	No certification plans for vendors Few supported versions available for vanilla / upstream Kubernetes. Most Organizations need a commercially supported version.	Mesos strength is in Big Data and analytics and not container Orchestration Complexity - does too much and is too generic - needs frameworks for most use cases	More aimed at developers than central IT Only available from Docker Inc.