Container Services

What we hear from developers

I need to create applications at a competitive rate without worrying about IT

New applications run smoothly on my machine but malfunction on traditional IT servers

My productivity and application innovation become suspended when I have to wait on IT

What we hear from IT

I need to manage servers and maintain compliance with little disruption

I’m unsure of how to integrate unfamiliar applications, and I require help from developers

I’m unable to focus on both server protection and application compliance

IT stress points

Security threats

Datacenter efficiency

Supporting innovation

Cloud is a new way to think about a datacenter

Traditional model

Dedicated infrastructure for each application

Purpose-built hardware

Distinct infrastructure and operations teams

Customized processes and configurations

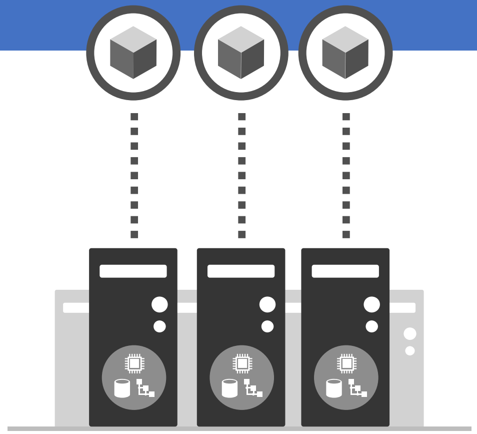
Cloud model

Loosely coupled apps and micro-services

Industry-standard hardware

Service-focused DevOps teams

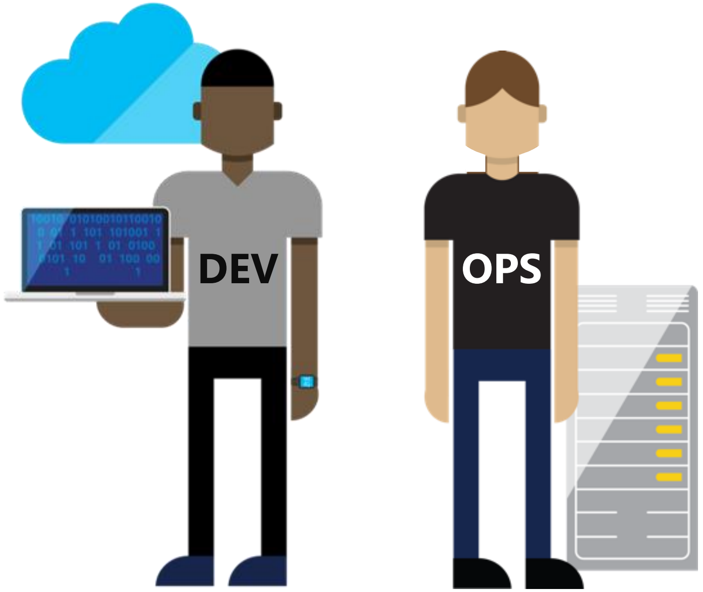
Standardized processes and configurations

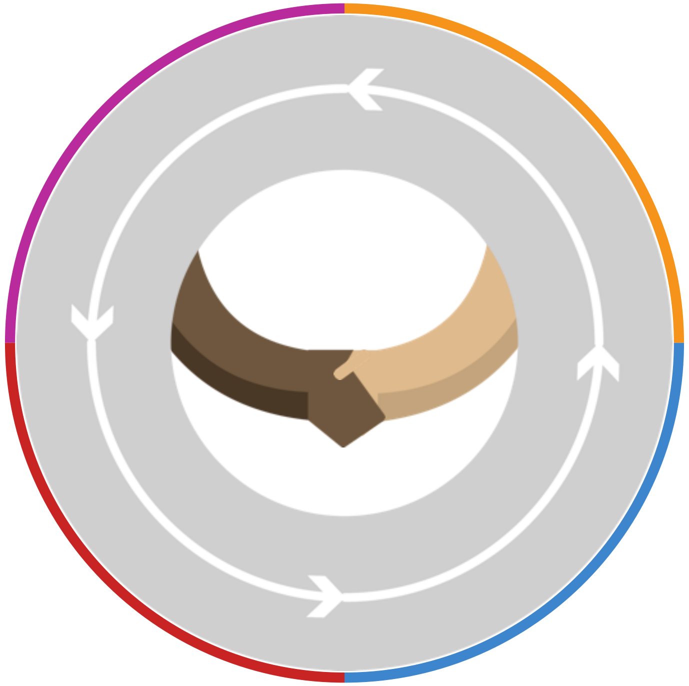


Servers

Services

DevOps: the three stage conversation



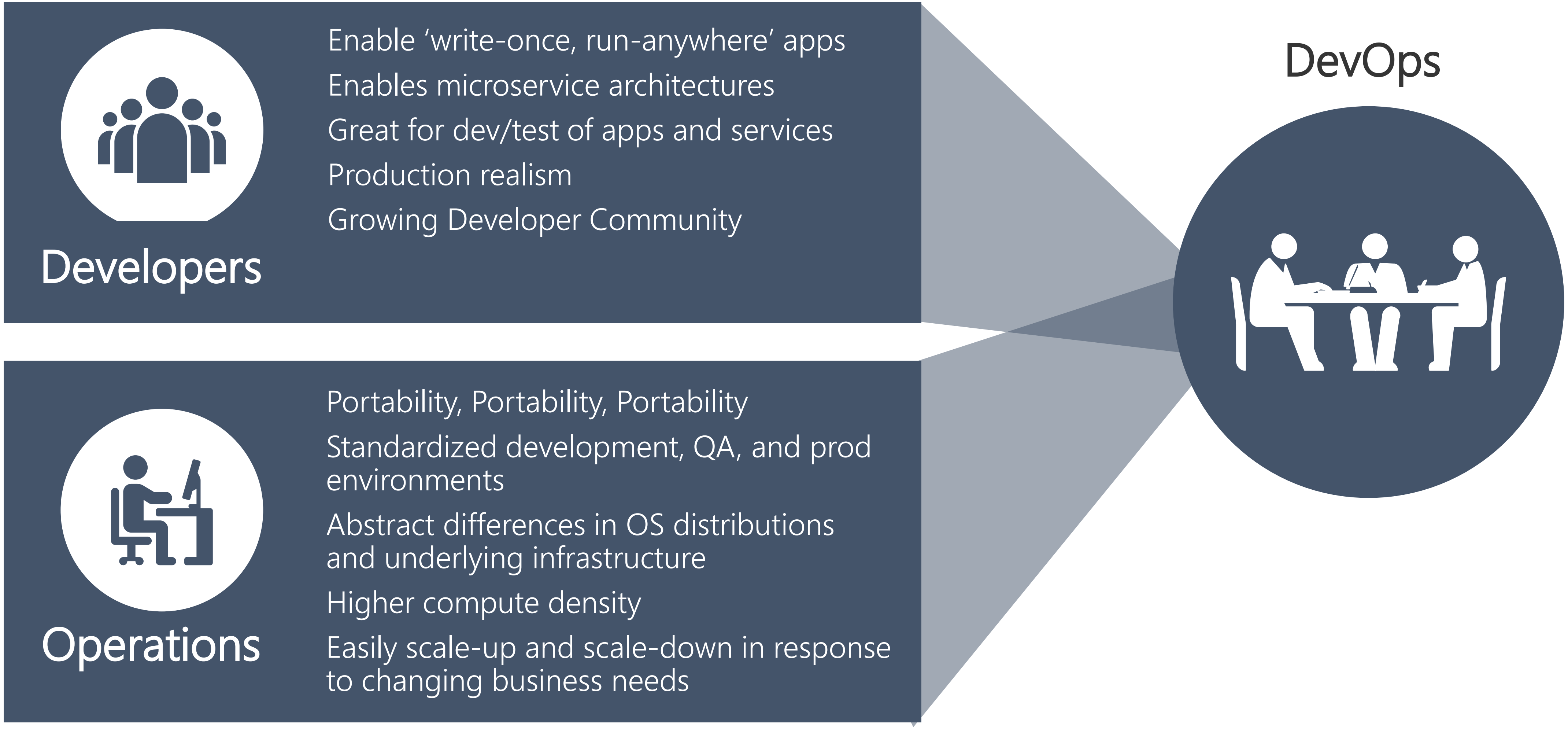


1 People

2 Process

3 Products

Why Containers?



Developers

- Enable ‘write-once, run-anywhere’ apps
- Enables microservice architectures
- Great for dev/test of apps and services
- Production realism
- Growing Developer Community

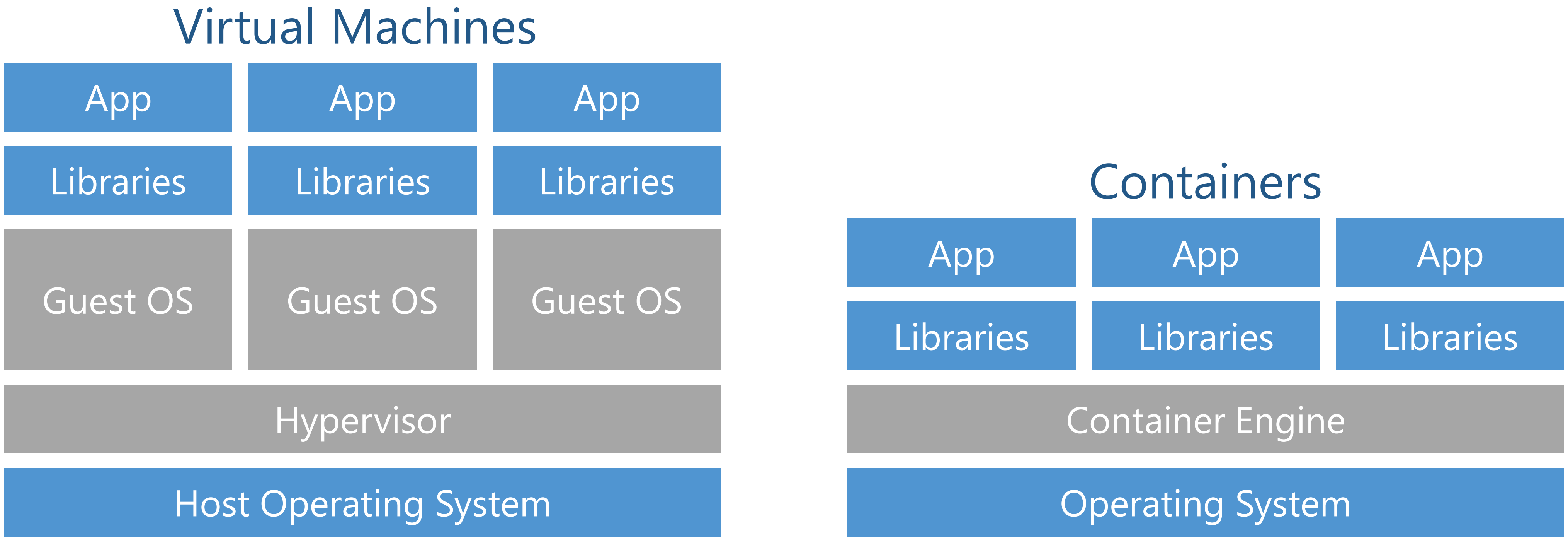
Operations

- Portability, Portability, Portability
- Standardized development, QA, and prod environments
- Abstract differences in OS distributions and underlying infrastructure
- Higher compute density
- Easily scale-up and scale-down in response to changing business needs

DevOps

Virtual Machine vs Containers

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



Virtual Machines

App App App

Libraries Libraries Libraries

Guest OS Guest OS Guest OS

Hypervisor

Host Operating System

Containers

App App App

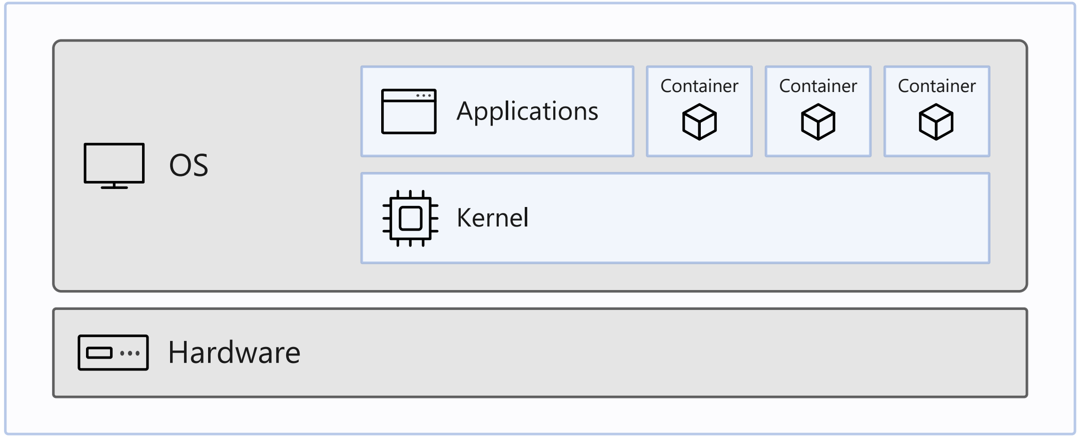
Libraries Libraries Libraries

Container Engine

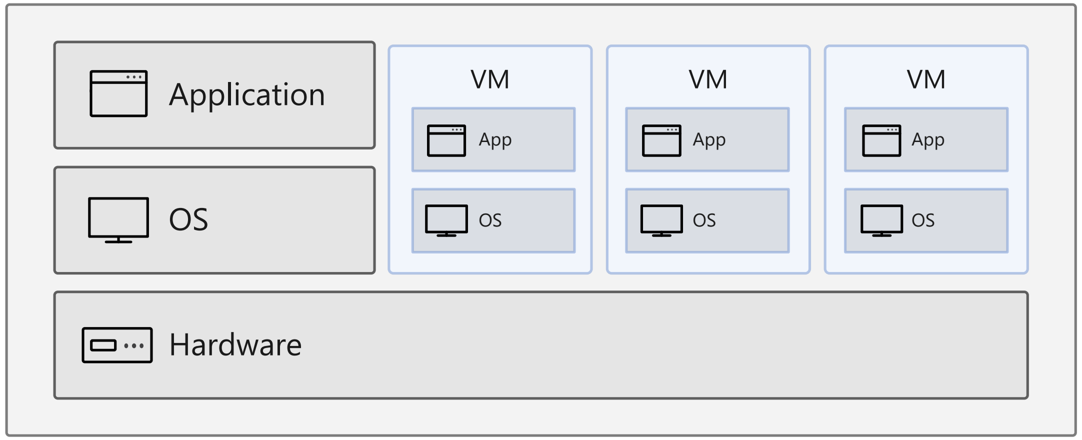
Operating System

What is a container?

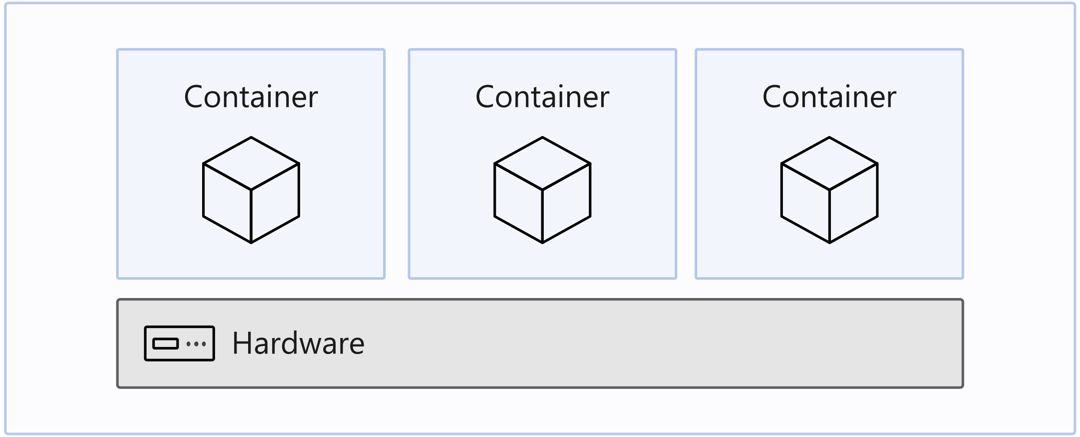
Containers = operating system virtualization



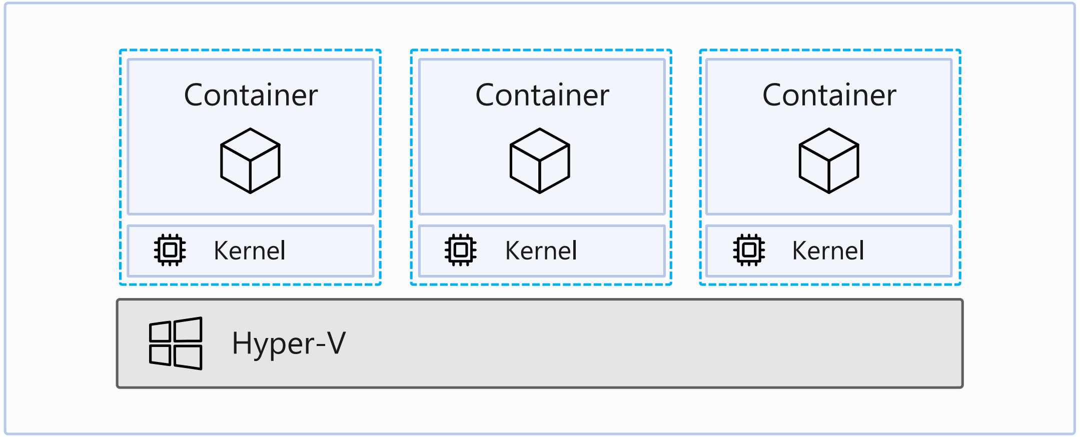
Traditional virtual machines = hardware virtualization



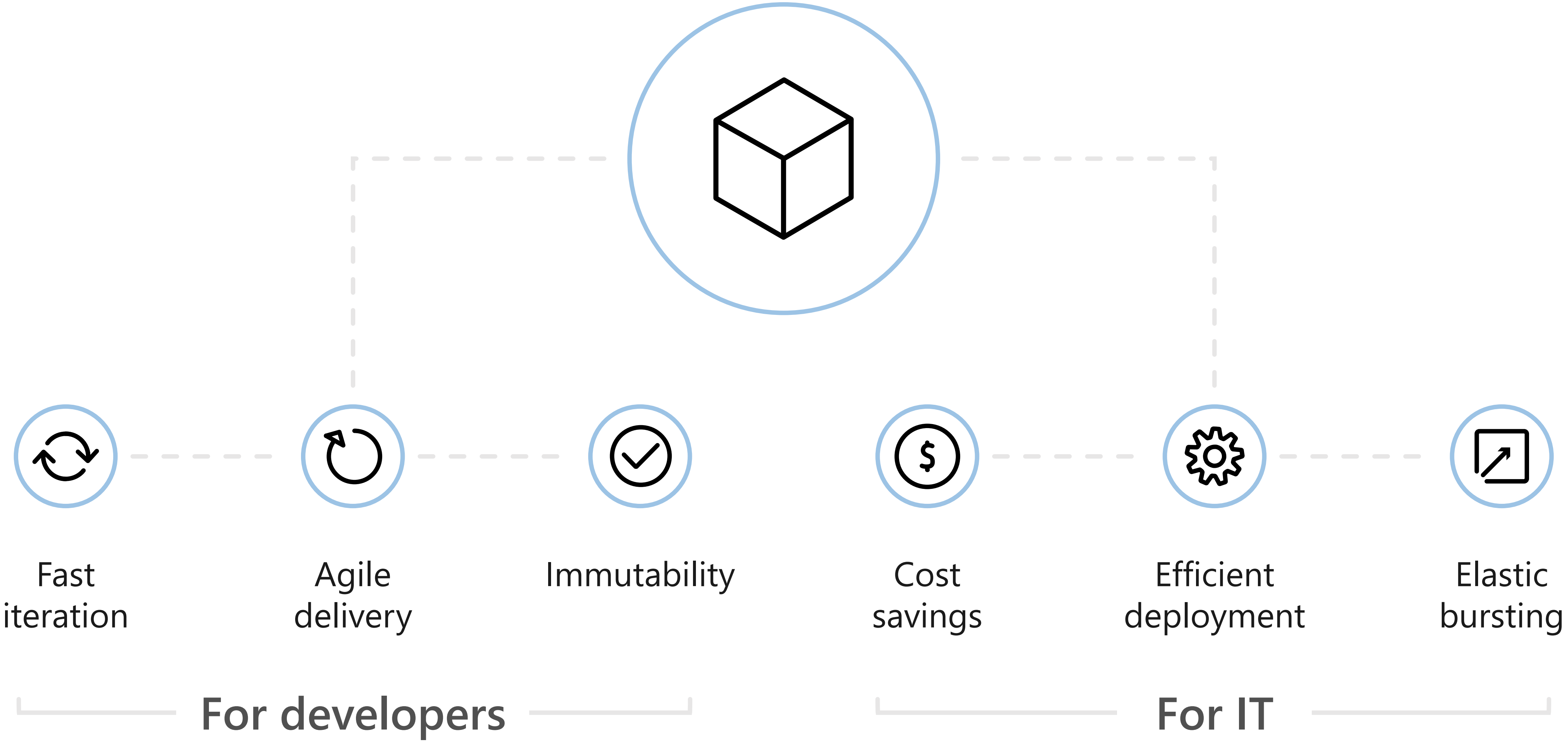
Windows Server containers: maximum speed and density



Hyper-V containers: isolation plus performance



The container advantage



Fast iteration

Agile delivery

Immutability

Cost savings

Efficient deployment

Elastic bursting

For developers

For IT

The elements of orchestration

Scheduling

Affinity/anti-affinity

Health monitoring

Failover

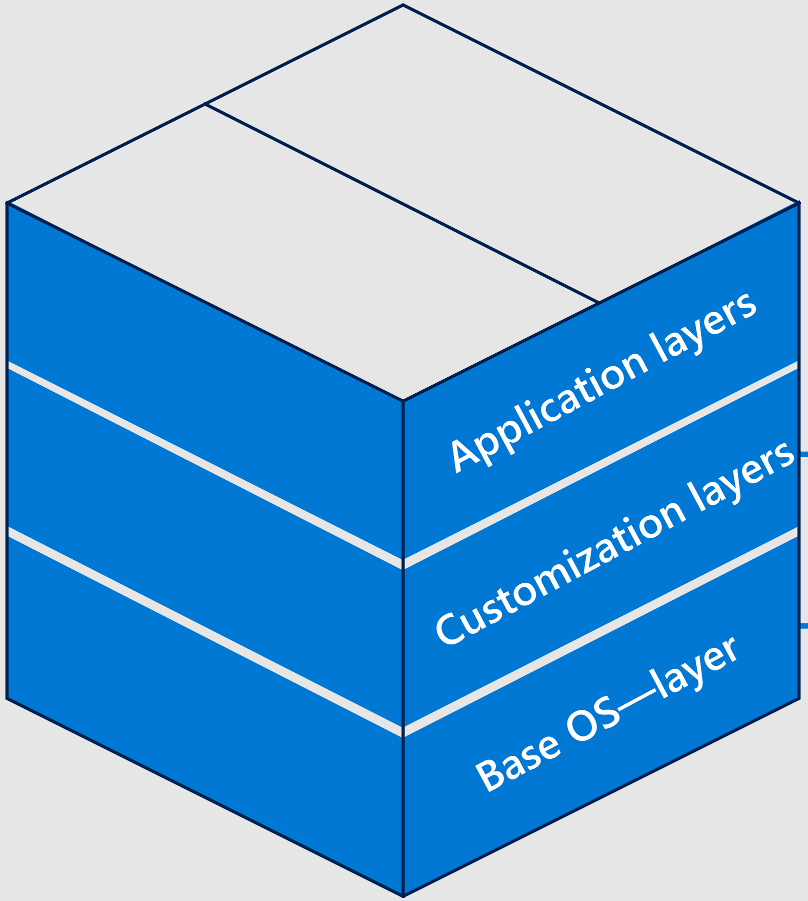
Scaling

Networking

Service discovery

Coordinated app upgrades

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

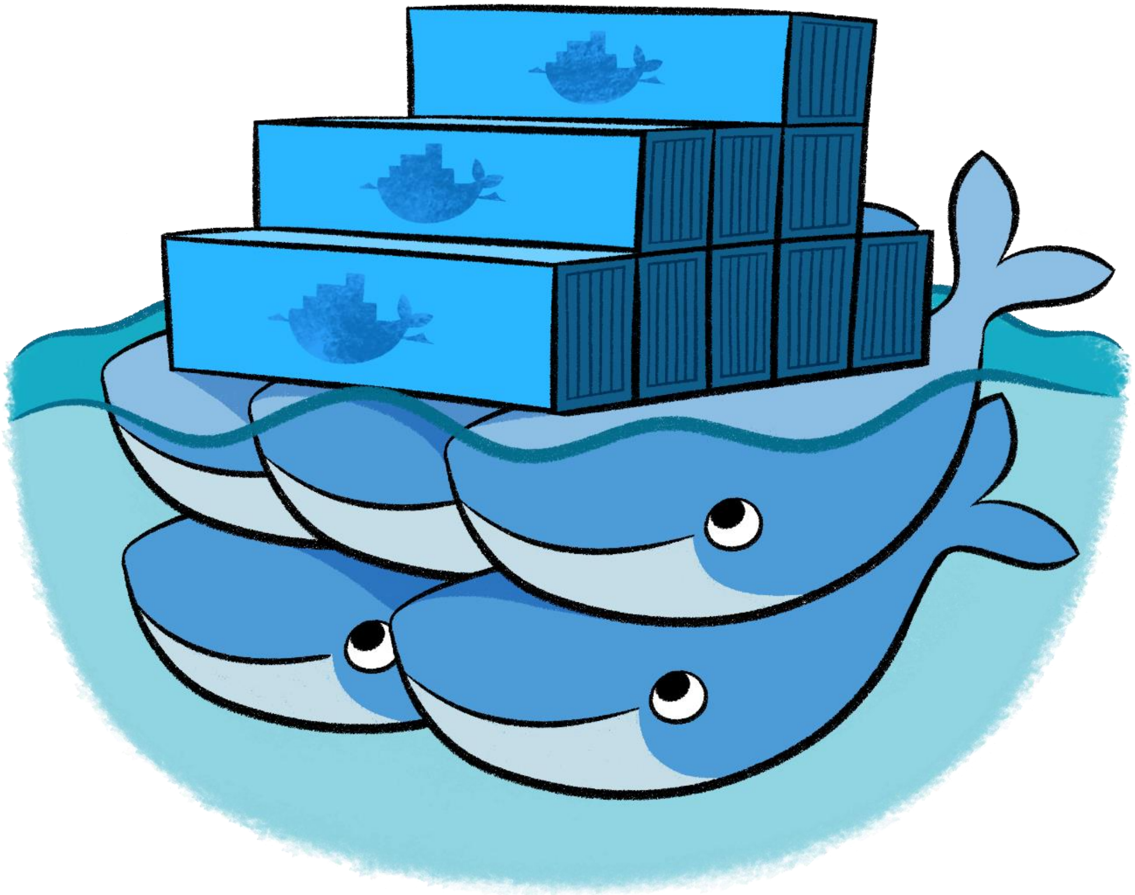
- Image size
  - Group related actions
  - Remove excess files
- Build speed
  - Multiple lines
  - Ordering of instructions
- Cosmetic optimizations

Docker

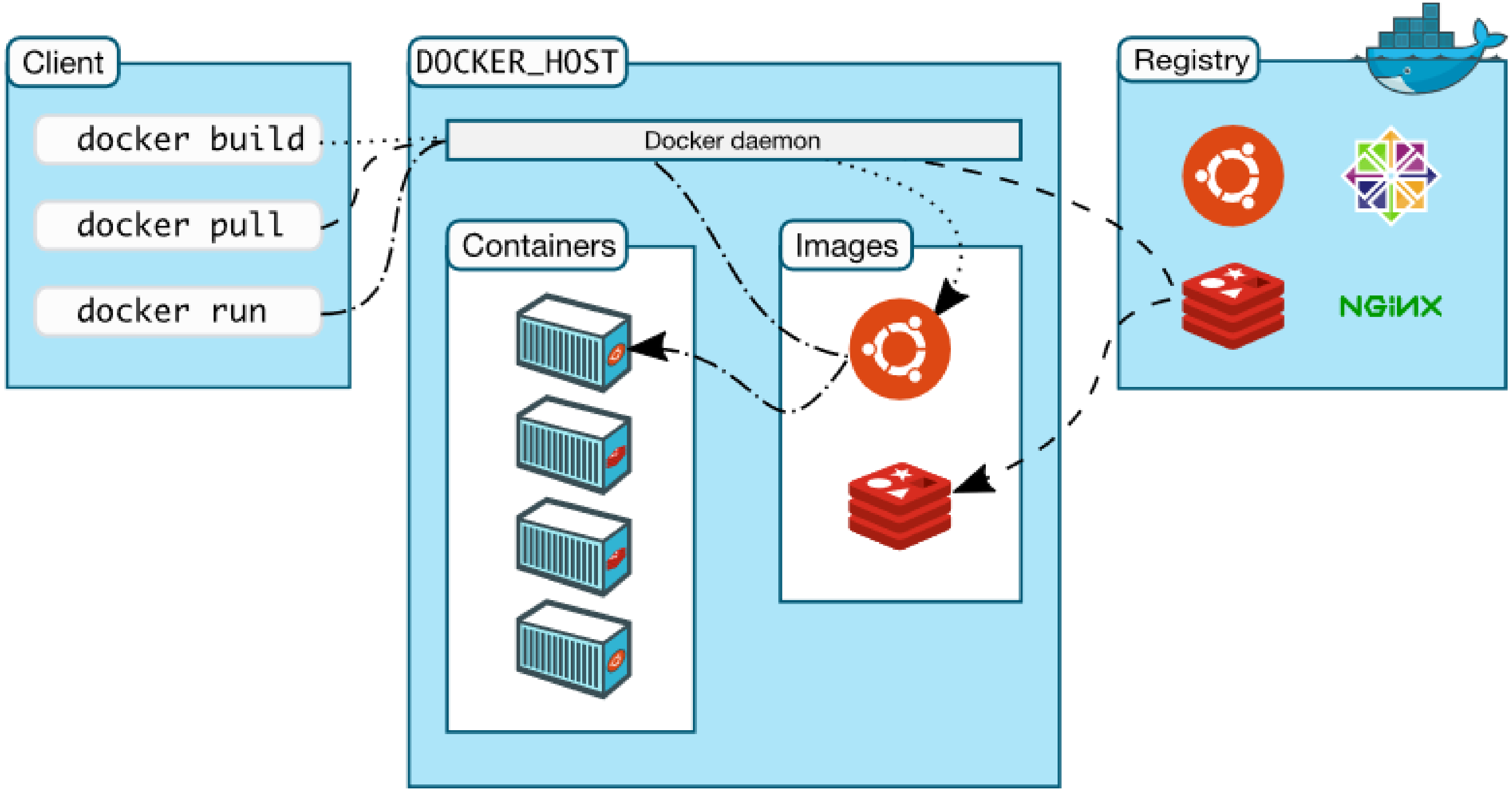
- Leading open-source containerization platform

  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

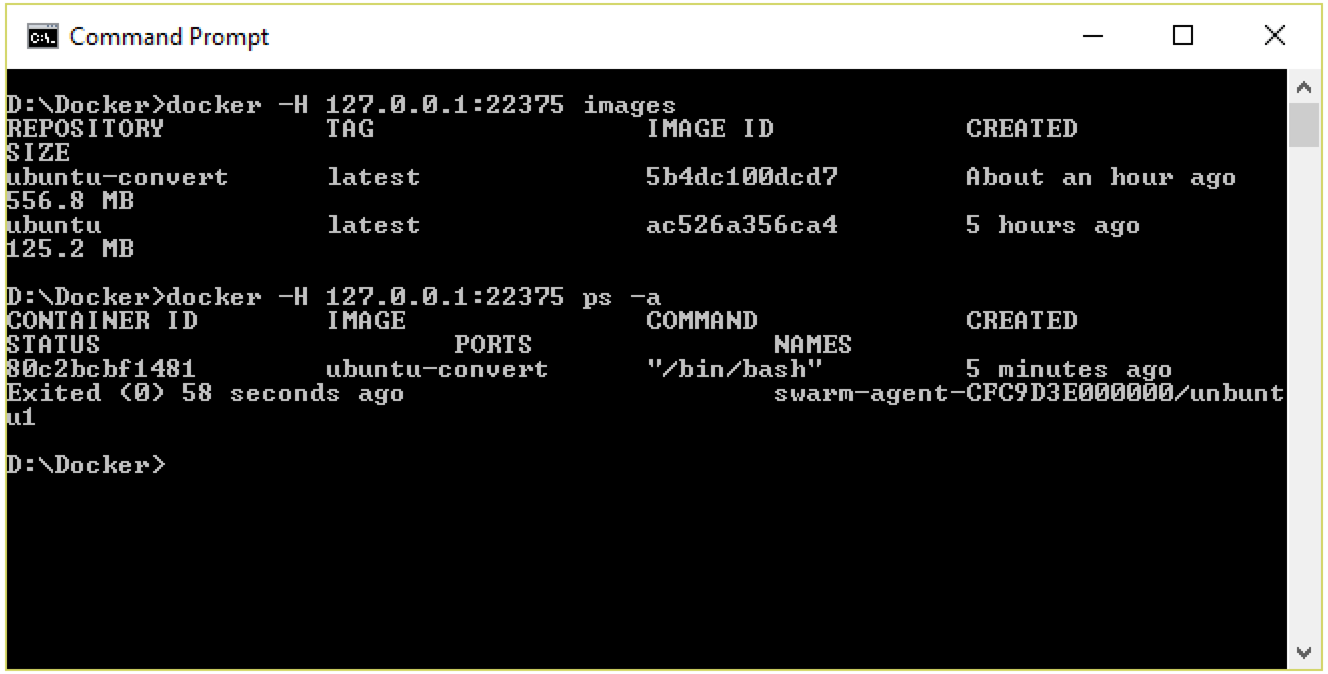


Docker Architecture



Docker CLI

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

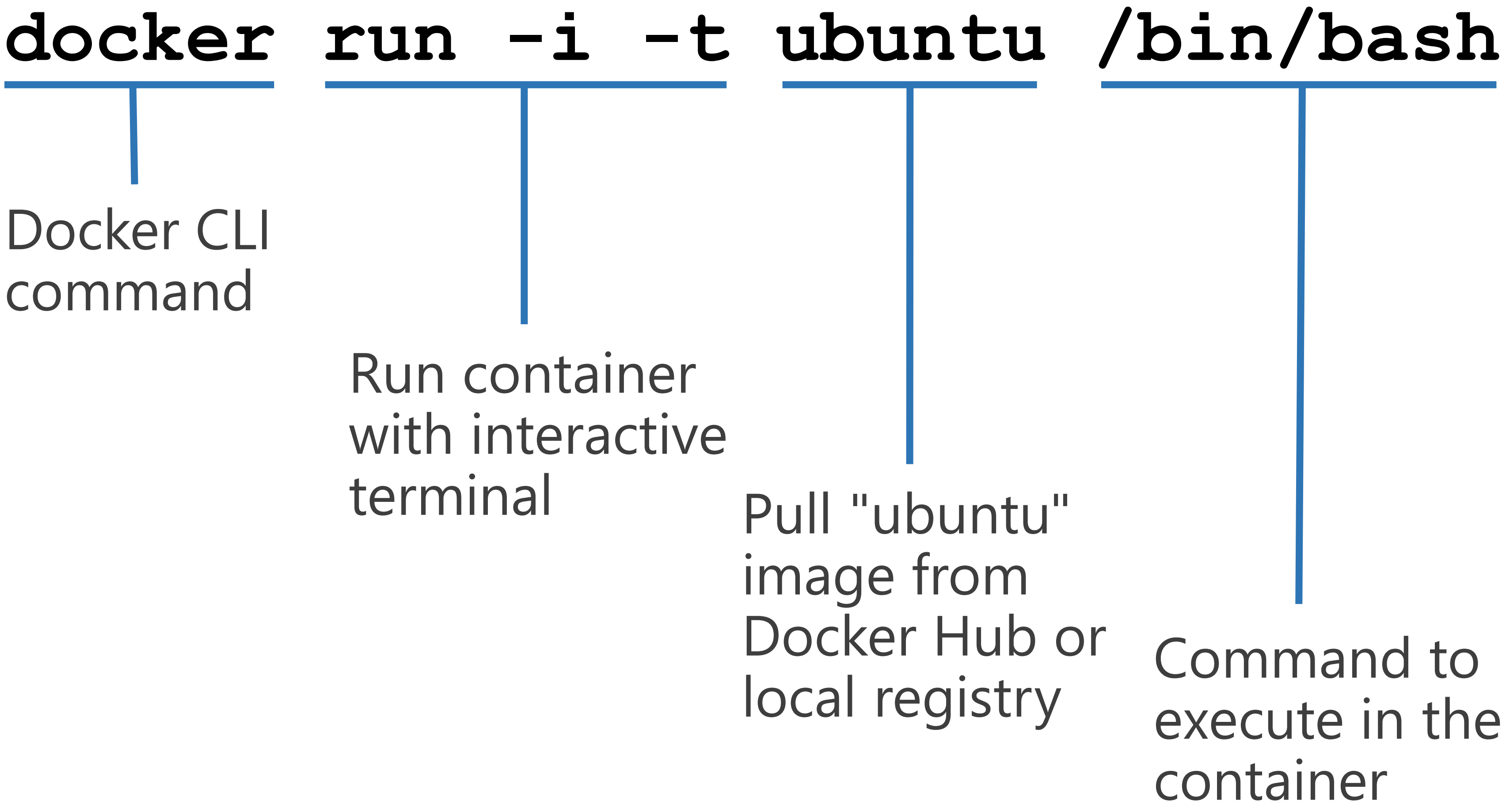


```
D:\Docker>docker -H 127.0.0.1:22375 images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
ubuntu-convert      latest              5b4dc100dcd7        About an hour ago
556.8 MB
ubuntu              latest              ac526a356ca4        5 hours ago
125.2 MB

D:\Docker>docker -H 127.0.0.1:22375 ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS                      PORTS               NAMES
80c2bcbf1481        ubuntu-convert      "/bin/bash"         5 minutes ago
Exited (0) 58 seconds ago                       swarm-agent-CFC9D3E000000/unbunt
u1

D:\Docker>
```

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

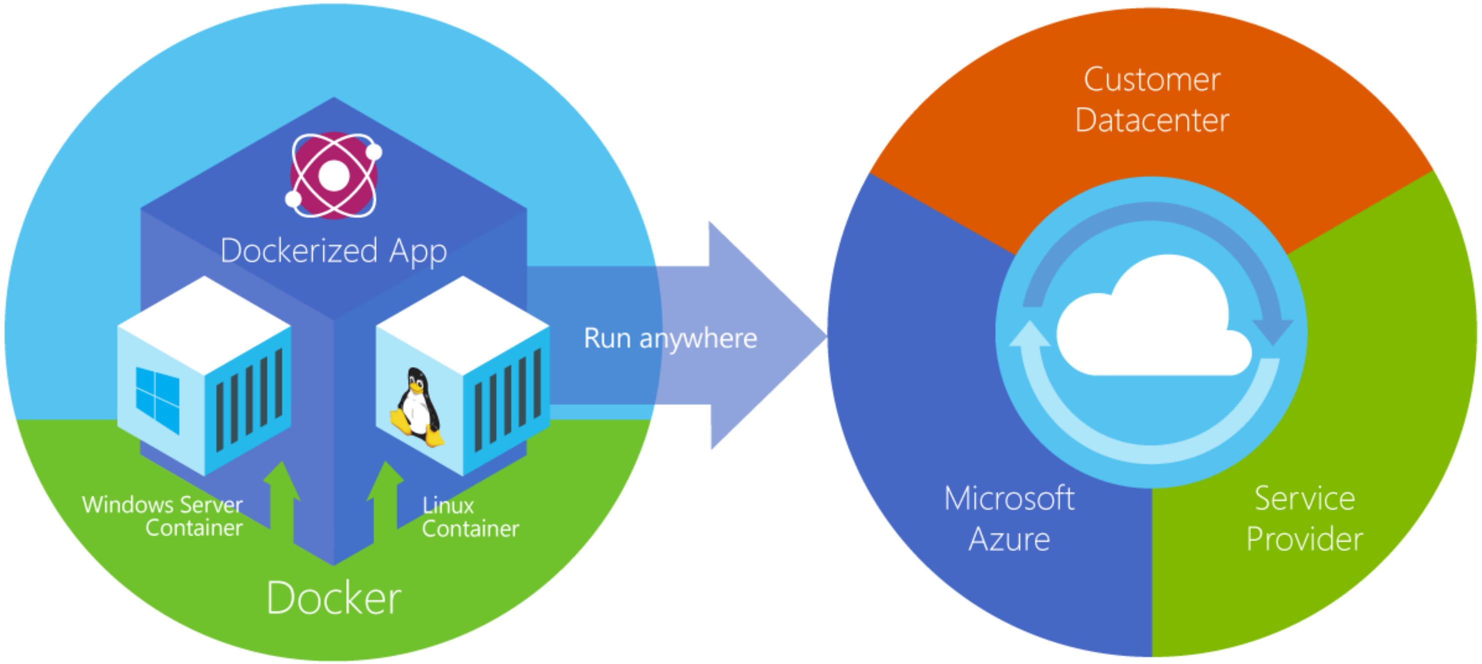
`docker ps` - List running Docker containers

`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)

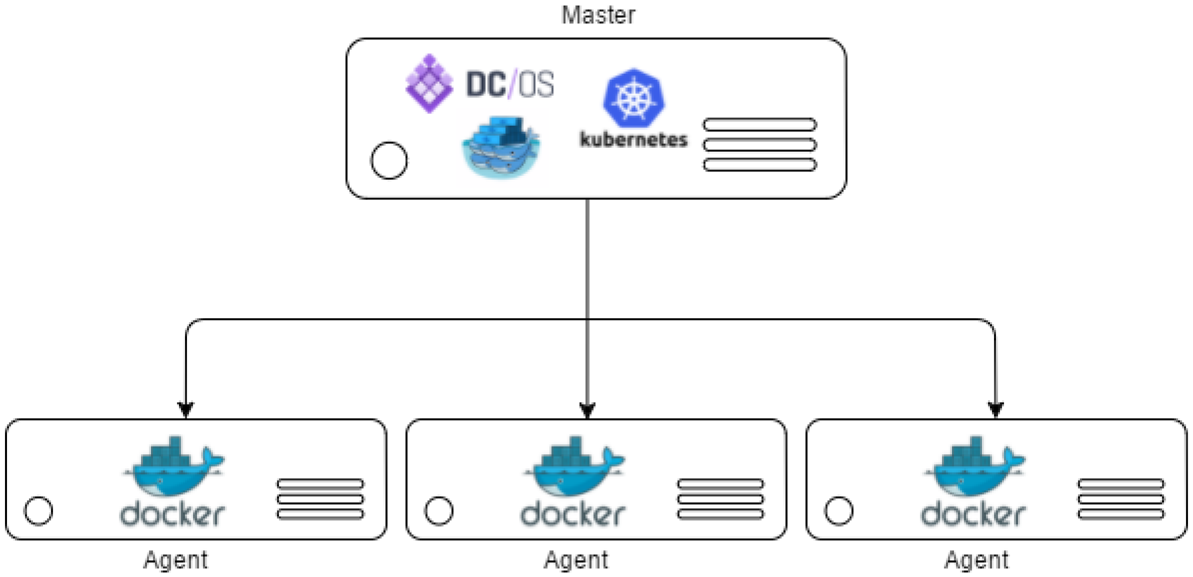


Container Orchestration

- 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

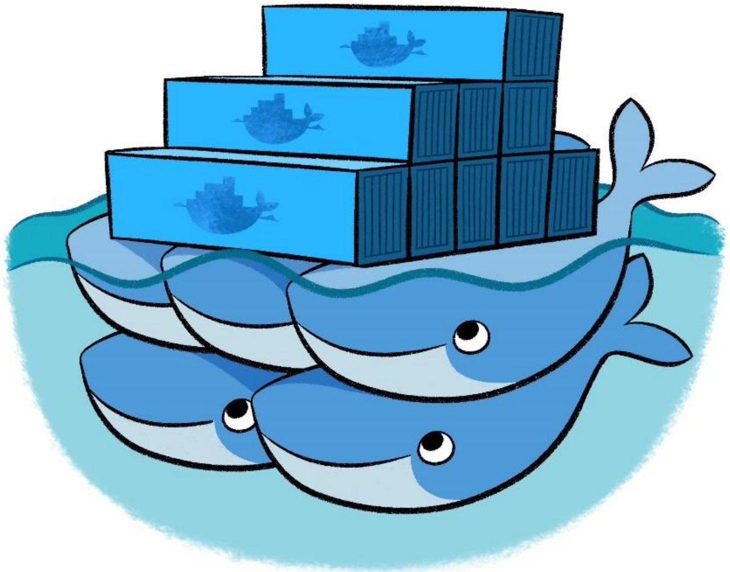
- Open-source orchestration engine from Google
- Provides a robust framework for container orchestration, yet remains lightweight and scalable
- Supported by Azure Container Service and tightly integrated with ACS, allowing Kubernetes to modify deployments

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

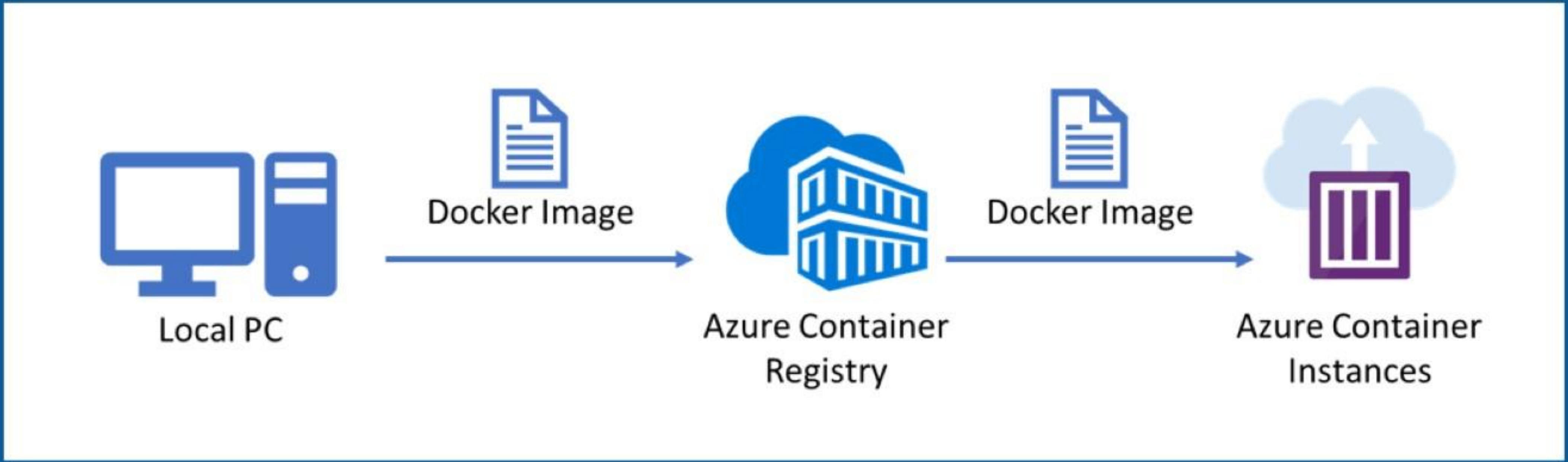
Docker Swarm

- 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

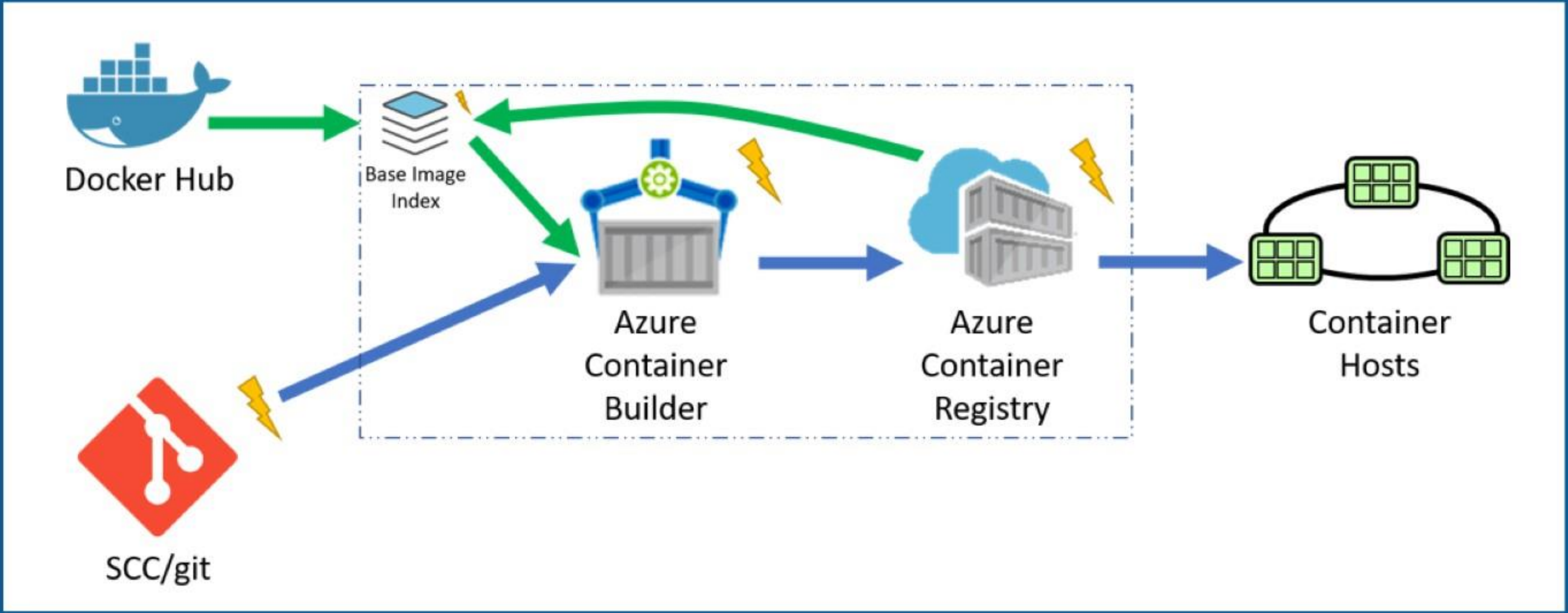
Premium

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

Image Storage

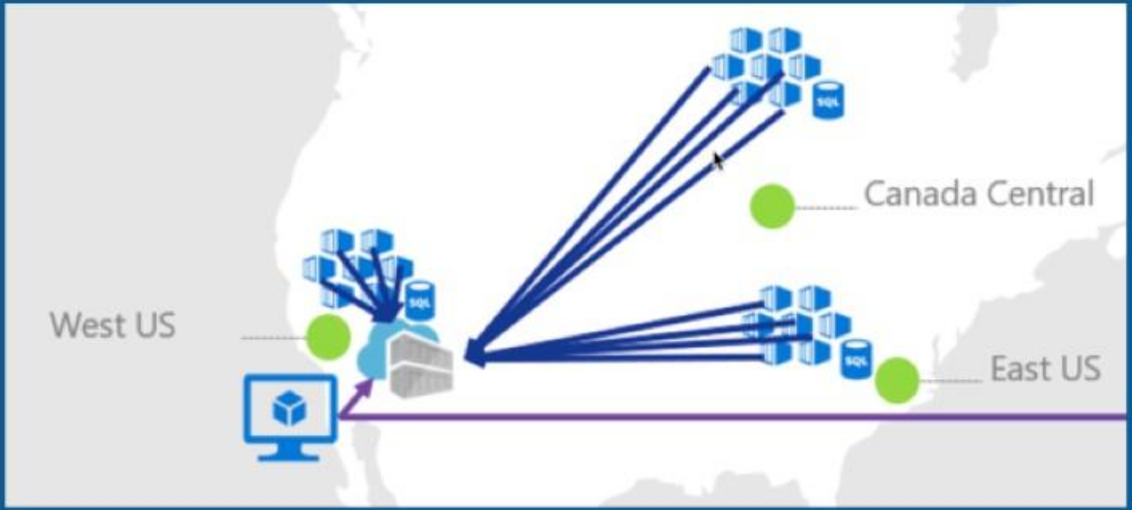
- All container images are encrypted at rest
- Encryption-at-rest for image data security
- Geo-redundancy for image data protection

Azure Container Registry Build Tasks

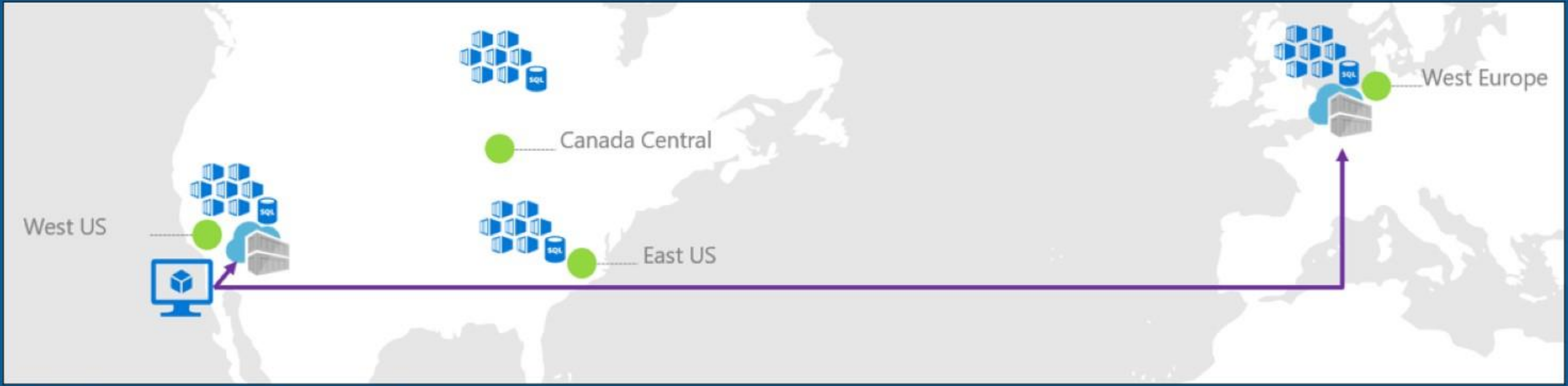


Geo-Replication

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

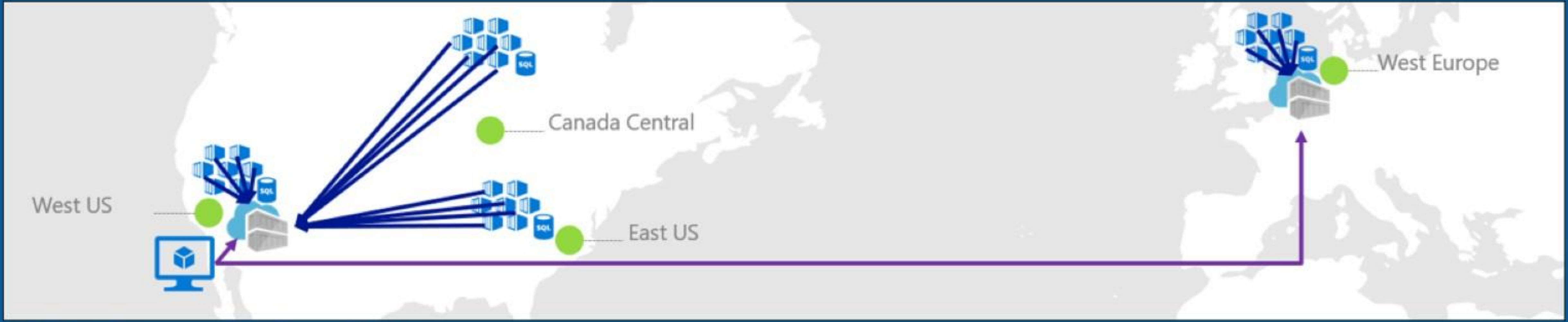


Geo-Replication Example Use Case



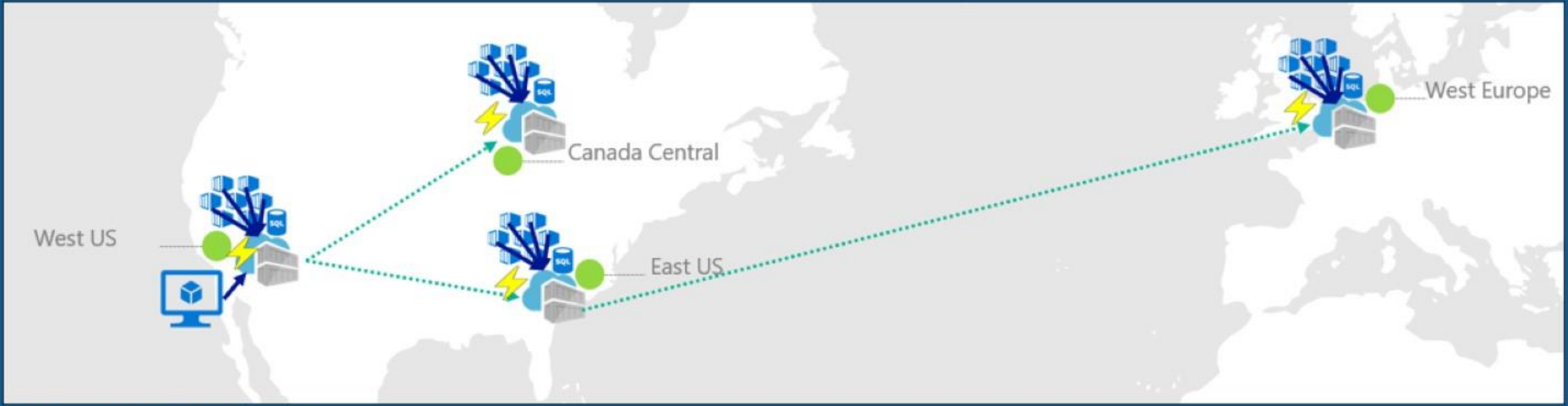
Pushing to multiple registries

Geo-Replication Example Use Case



Pulling from multiple registries

Geo-Replication Example Use Case



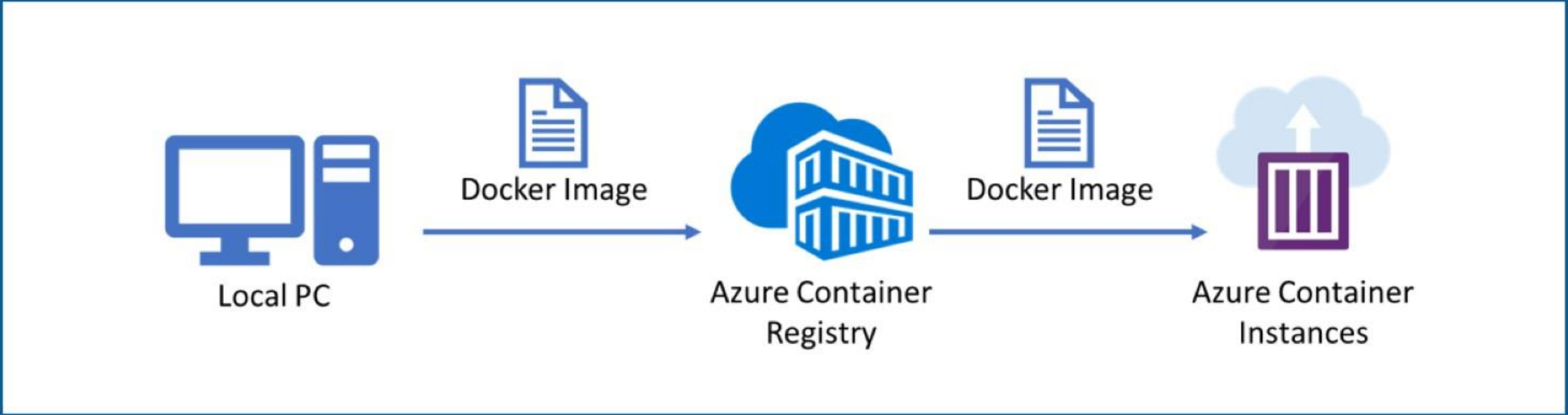
Using Geo-Replication

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

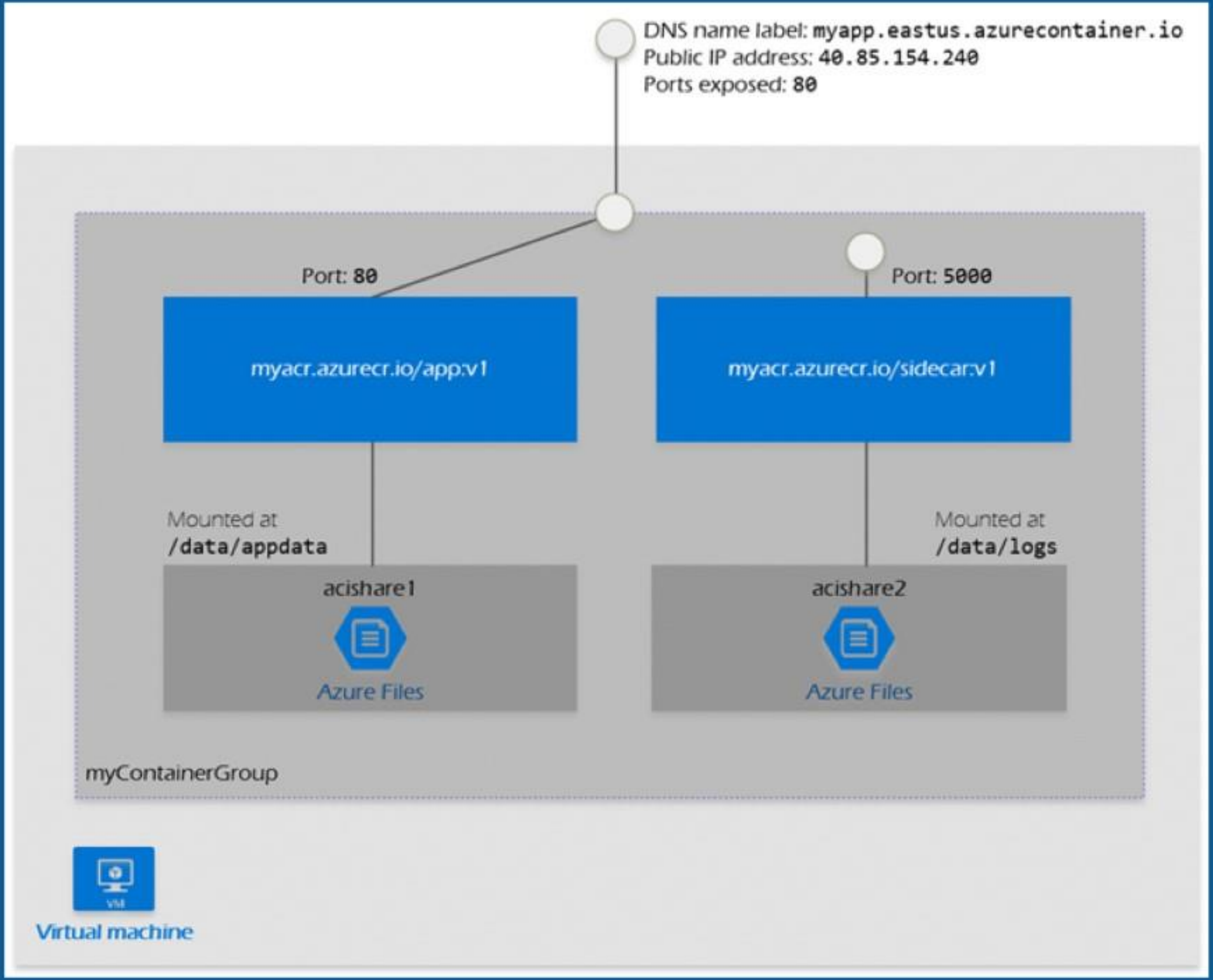
Persistent storage

Co-scheduled groups

Container Orchestrators

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

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

Azure Kubernetes Service

What is Kubernetes

kubernetes

•

Kubernetes = **K8s** .8 for the 8 letters between K and s •

•

- Born at Google , and they use K8s to run their data center
- Donated in 2014
- Written in Go/Golang

•

- Kubernetes means “governor” or “captain” in Greek. The symbol is the wheel of the ship .
- **Documentation : https://kubernetes.io/docs/home.**

Features of Kubernetes

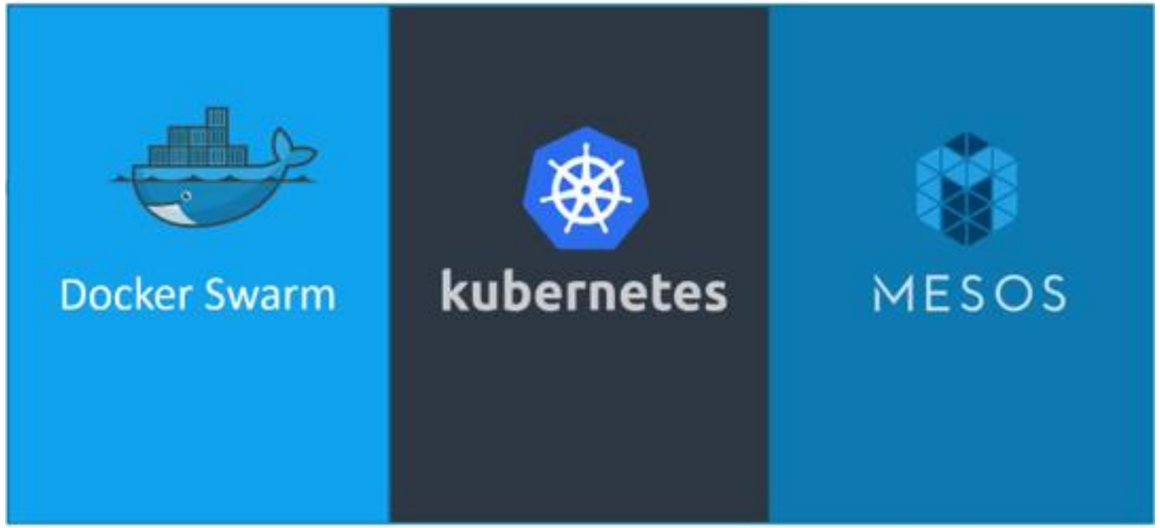
Open source container orchestrator that automates deployment, scaling, and management of applications

Features include:

- Automatic bin-packing
- Self-healing
- Horizontal scaling
- Service discovery and load balancing
- Automated rollouts and rollbacks
- Secret and configuration management
- Storage orchestration
- Batch execution

Orchestration technologies

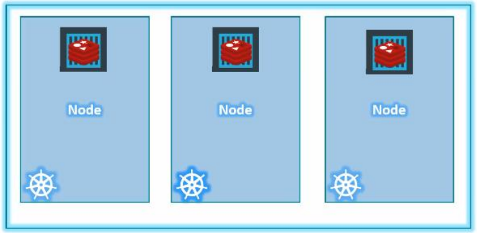
- Docker Swarms easy to set up and get started
- Kubernetes most popular , a bit difficult to set up and get started , but provide a lot of option to customize deployments and complex architecture . One of the Top ranked projects in GitHub. Supported on all the public clouds providers : GCP, AWS , Azure and on promise
- MESOS from Apache support many advanced features but quiet difficult to set up and get started .



Cluster Components

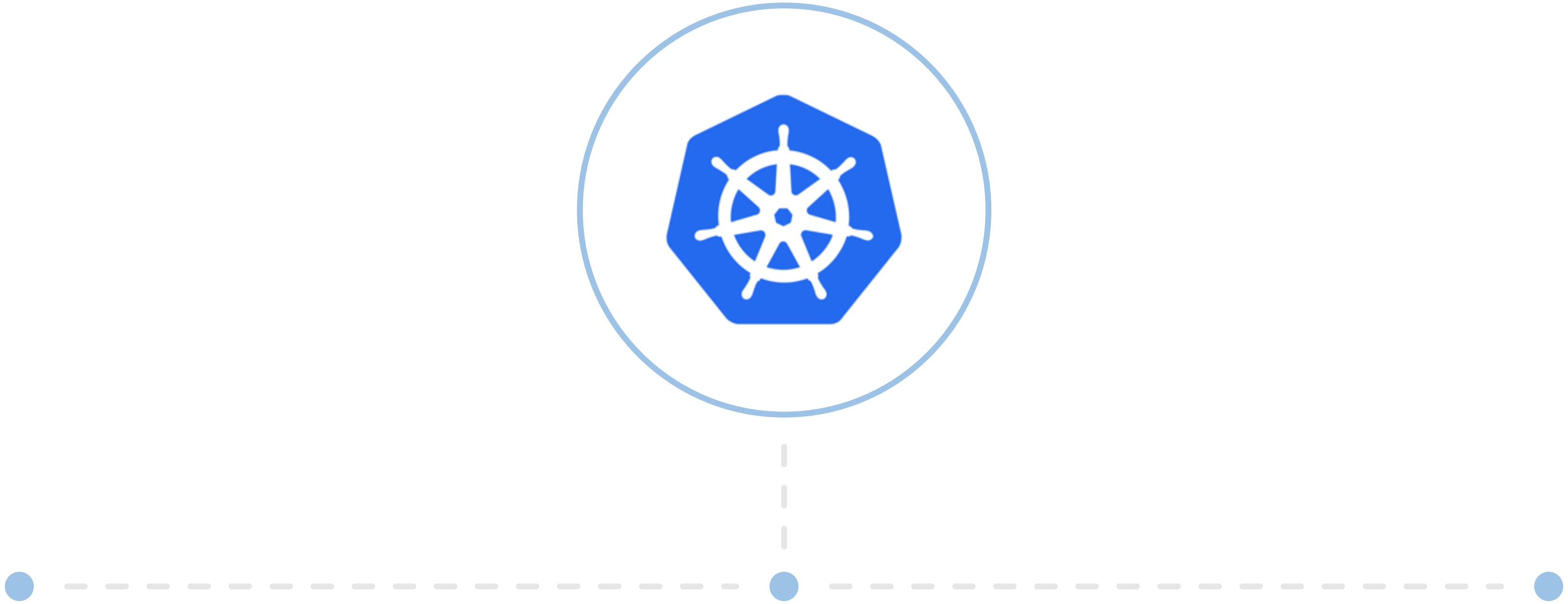
kubernetes

- For the purposes of SQL Server with k8s here are the most significant :
- **<u>Cluster</u>**
- **A k8s cluster is a deployment of containers in pods to a set of nodes.**
- **More than one node for redundancy and sharing loads .**



- **<u>Node:</u>**
- **A worker machine in Kubernetes, previously known as minion.**
- **(https://kubernetes.io/docs/concepts/architecture/nodes/) . A node may be a VM or physical machine**.

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

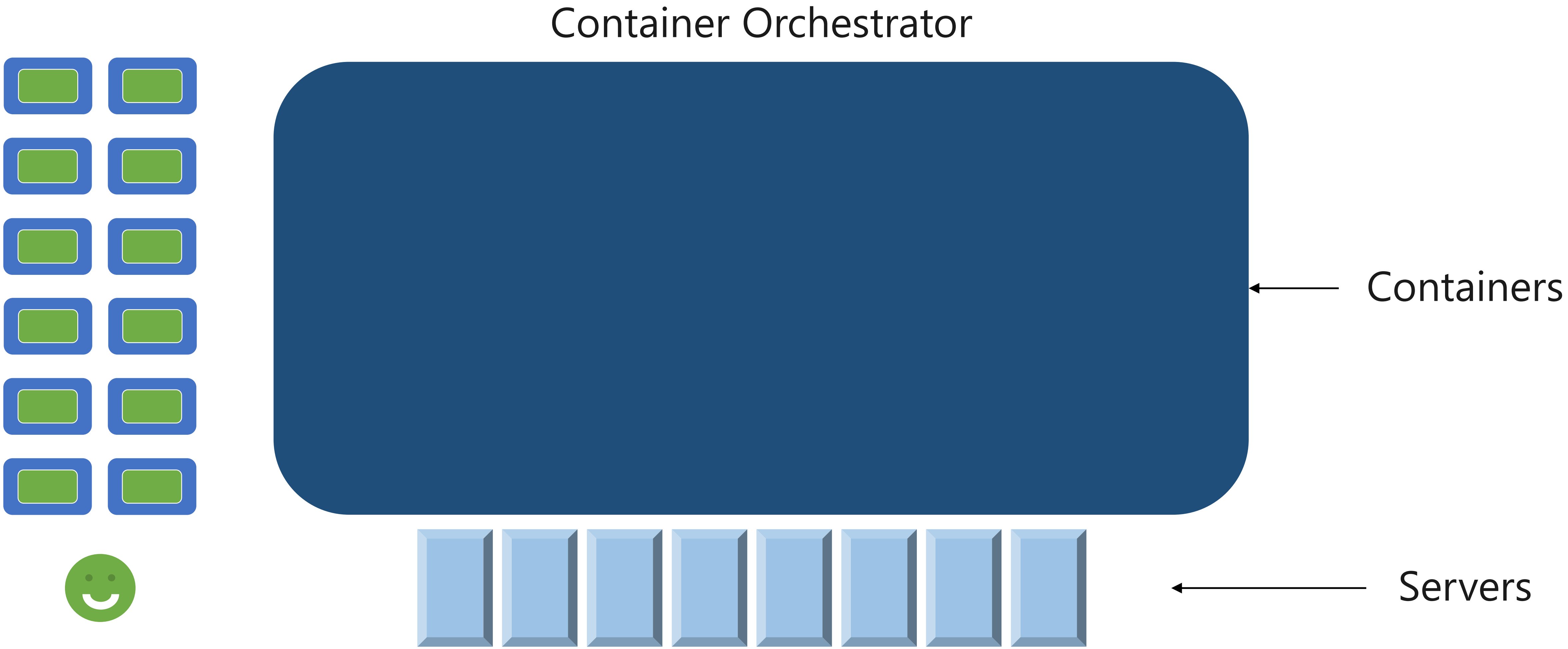
Deploy your
applications quickly
and predictably

Scale your
applications on
the fly

Roll out
new features
seamlessly

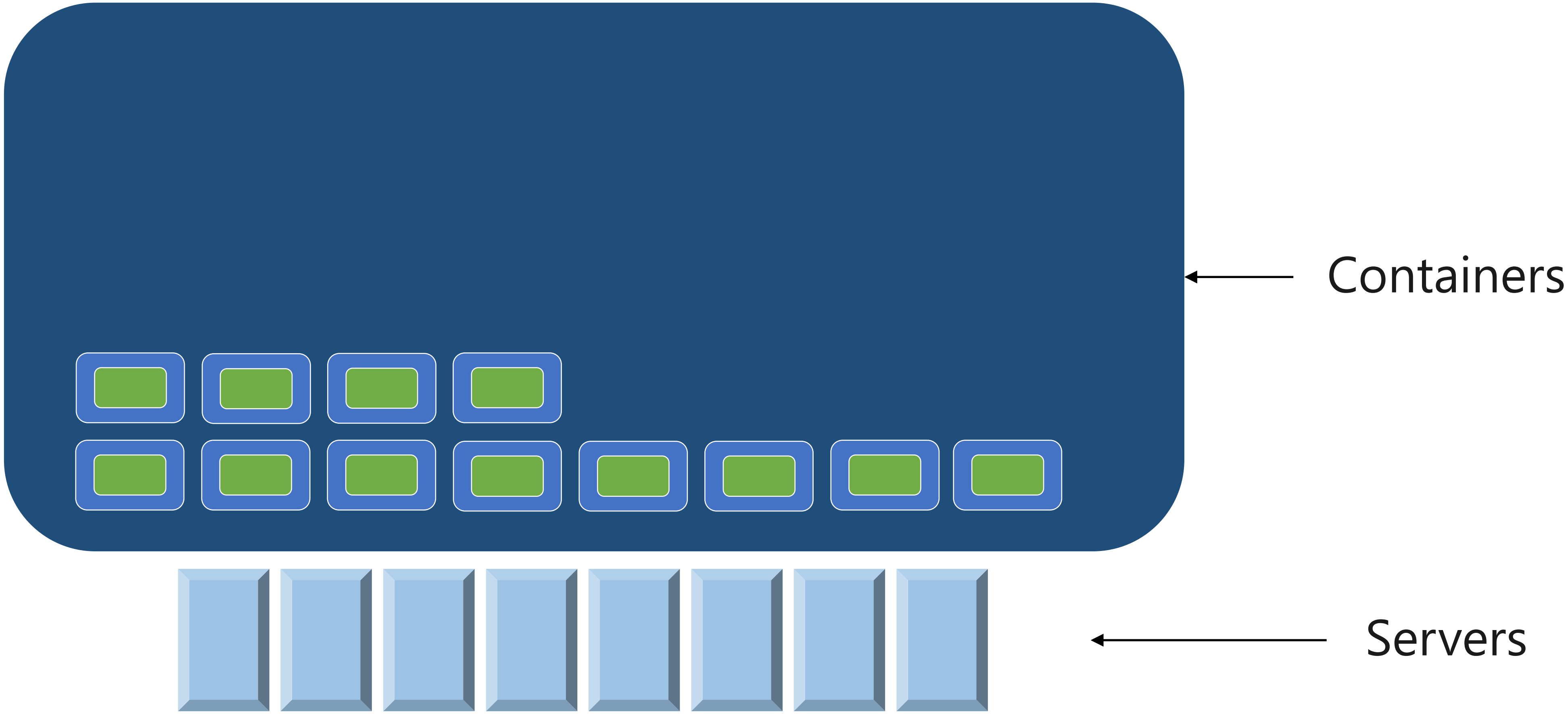
Limit hardware
usage to required
resources only

Kubernetes - Agility



Kubernetes - Agility

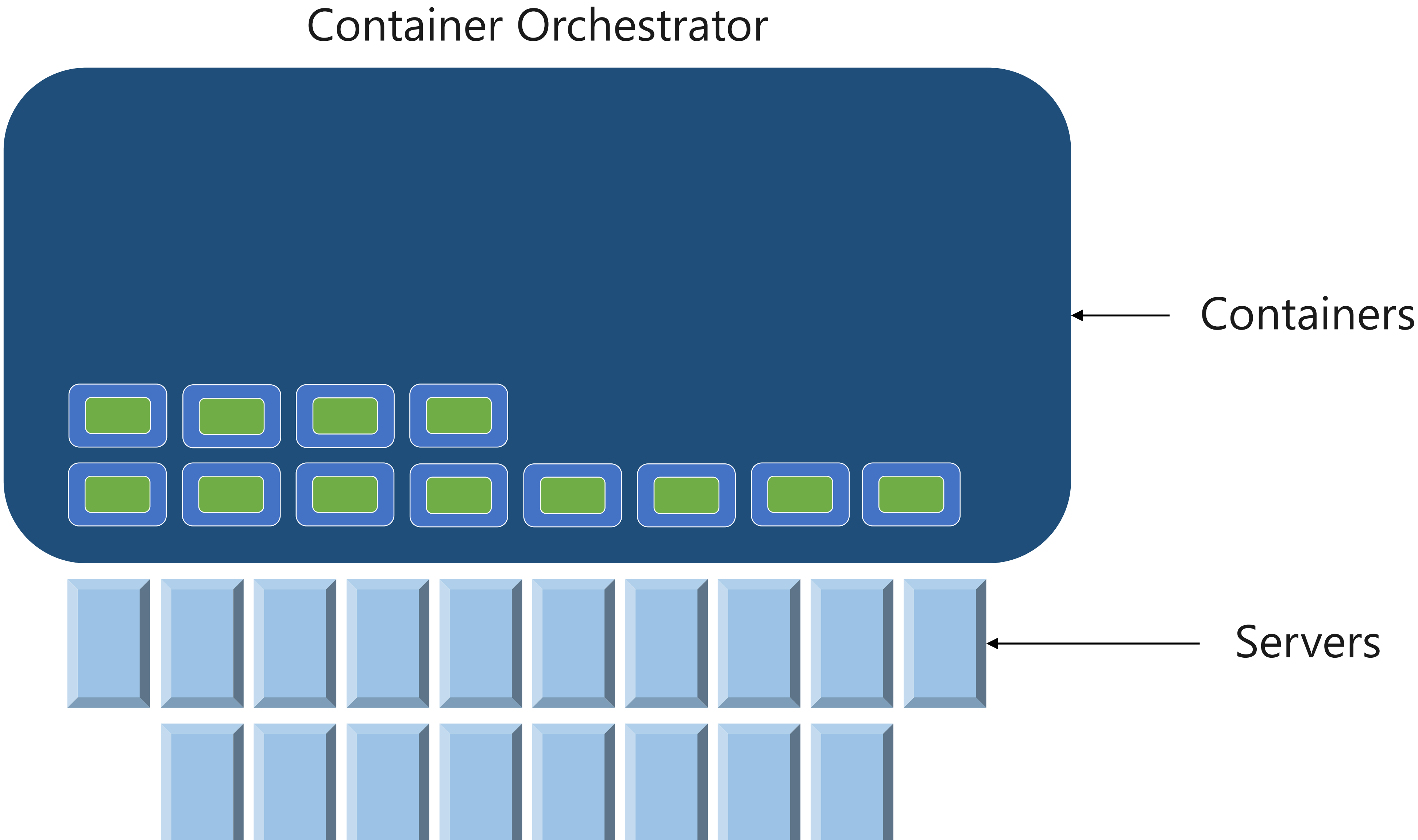
Container Orchestrator



Containers

Servers

Kubernetes - Scalability

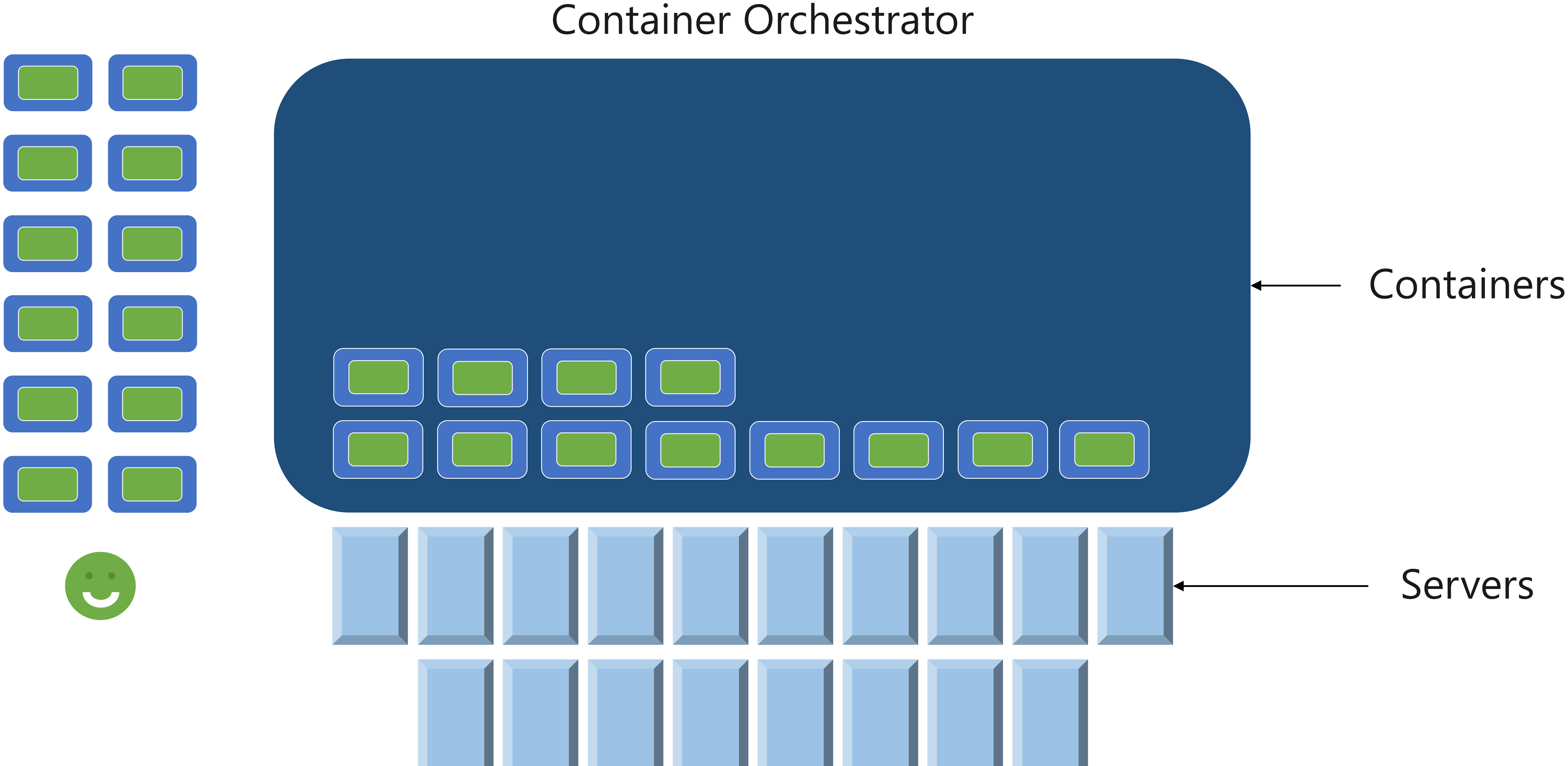


Container Orchestrator

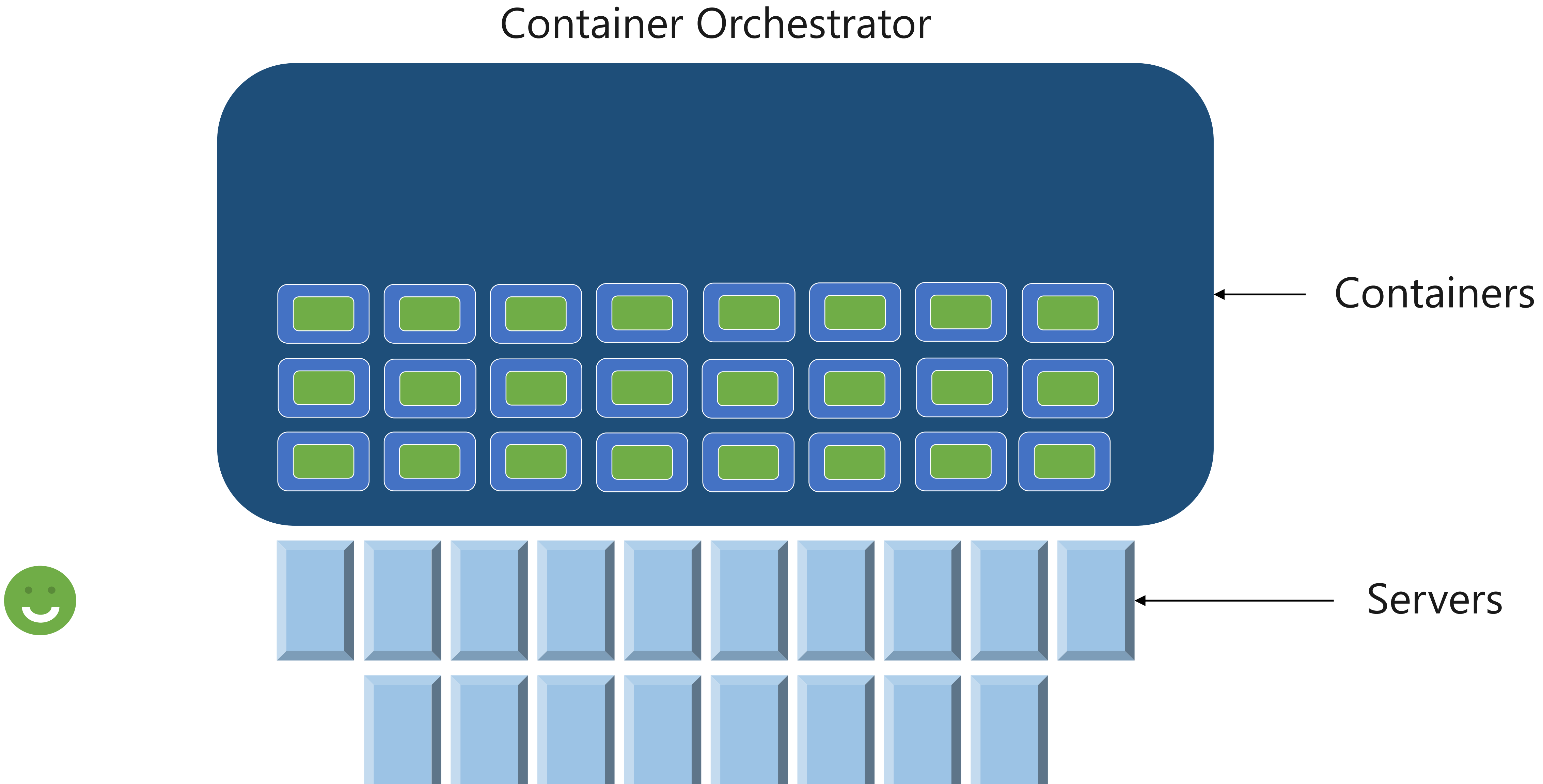
Containers

Servers

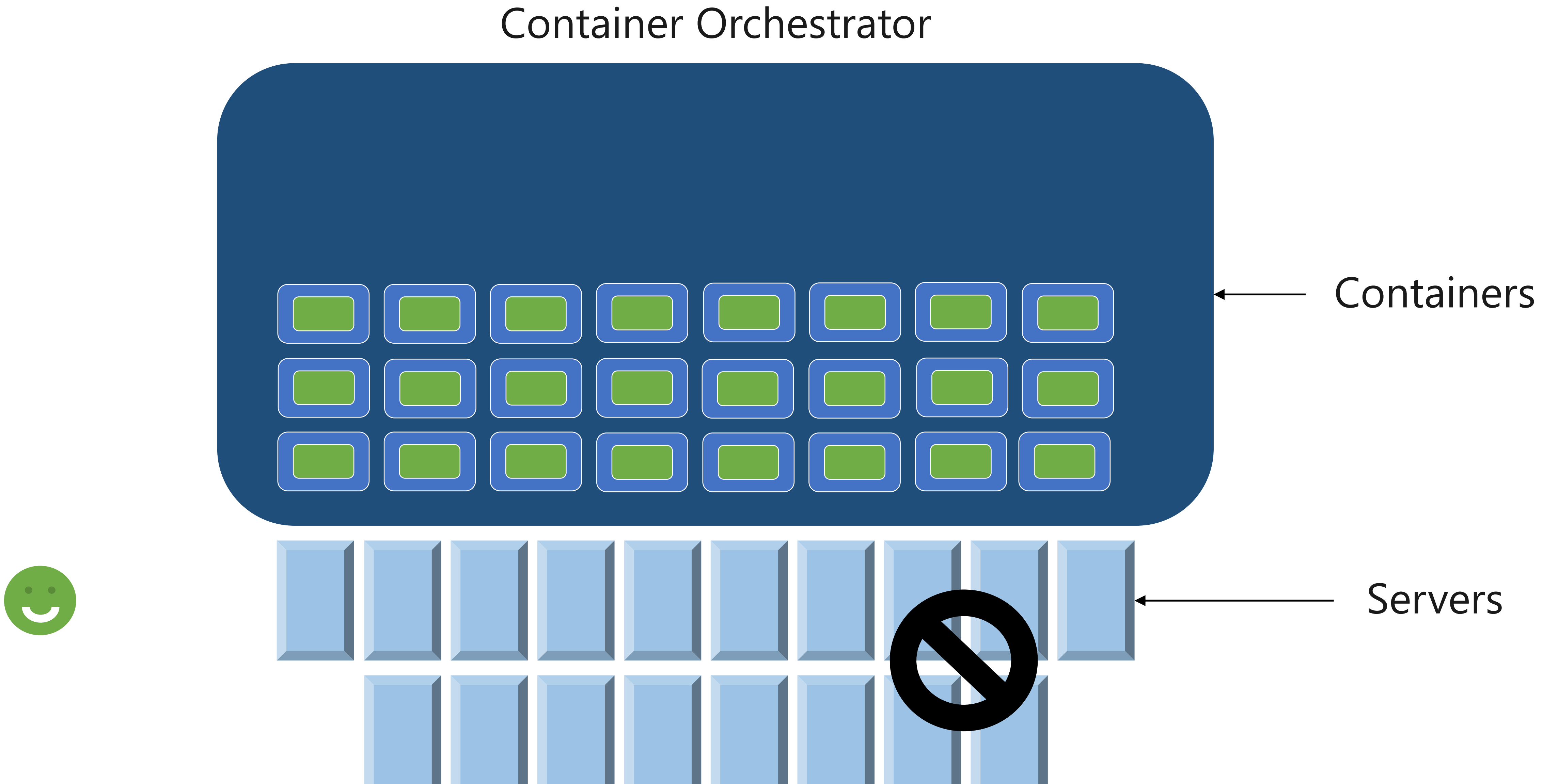
Kubernetes - Scalability



Kubernetes - Scalability



Kubernetes - Reliability



Where can I get/run Kubernetes

- Minikube
- Docker Edge
- Cloud
  - Azure
    - ACS
    - AKS
    - ACI
  - AWS
    - EKS
  - GCP/GKE

Why AKS?

Easy to use

Fastest path to Kubernetes on Azure
Up and running with 3 simple commands

Easy to manage

Automated upgrades and patching
Easily scale the cluster up and down
Self-healing control plane

Uses Open APIs

100% upstream Kubernetes

Getting Started with AKS

```
$ az aks create -g myResourceGroup -n myCluster --generate-ssh-keys
\ Running ..

$ az aks install-cli
Downloading client to /usr/local/bin/kubectl ..

$ az aks get-credentials -g myResourceGroup -n myCluster
Merged "myCluster" as current context ..

$ kubectl get nodes
NAME                       STATUS    AGE       VERSION
aks-mycluster-36851231-0   Ready     4m        v1.8.1
aks-mycluster-36851231-1   Ready     4m        v1.8.1
aks-mycluster-36851231-2   Ready     4m        v1.8.1
```

Managing an AKS cluster

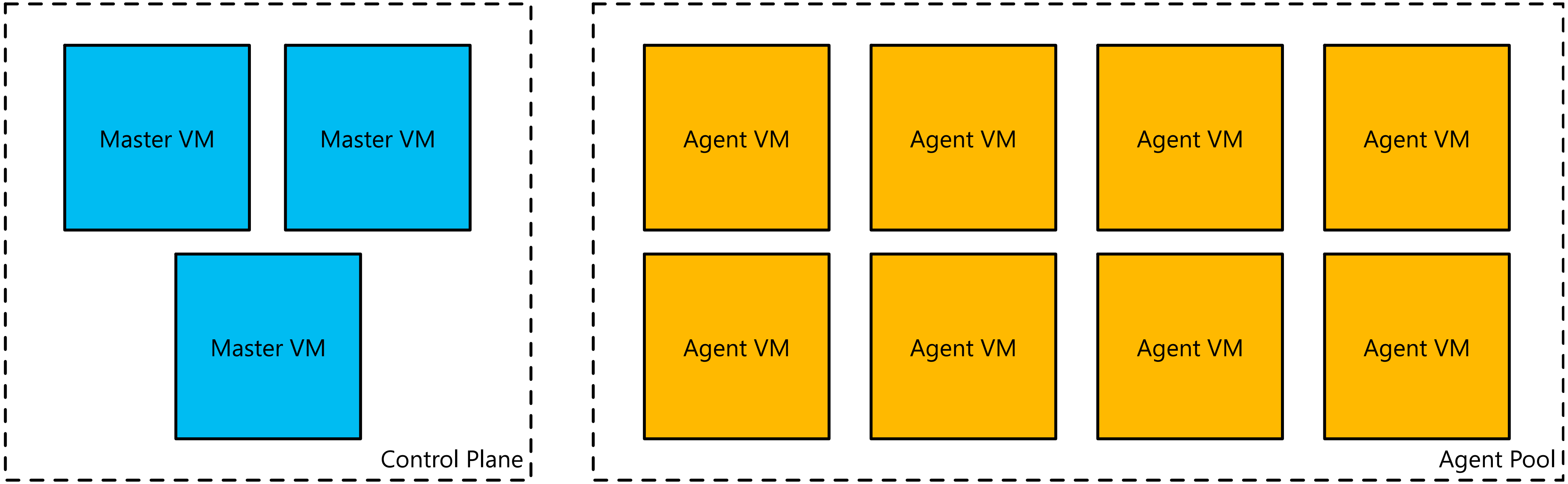
```
$ az aks list –o table
Name                Location    ResourceGroup     KubernetesRelease  ProvisioningState
------------------  ----------  --------------  -------------------  -------------------
myCluster           westus2     myResourceGroup                1.7.7  Succeeded
```

```
$ az aks upgrade -g myResourceGroup -n myCluster ––kubernetes-version 1.8.1
\ Running ..
```

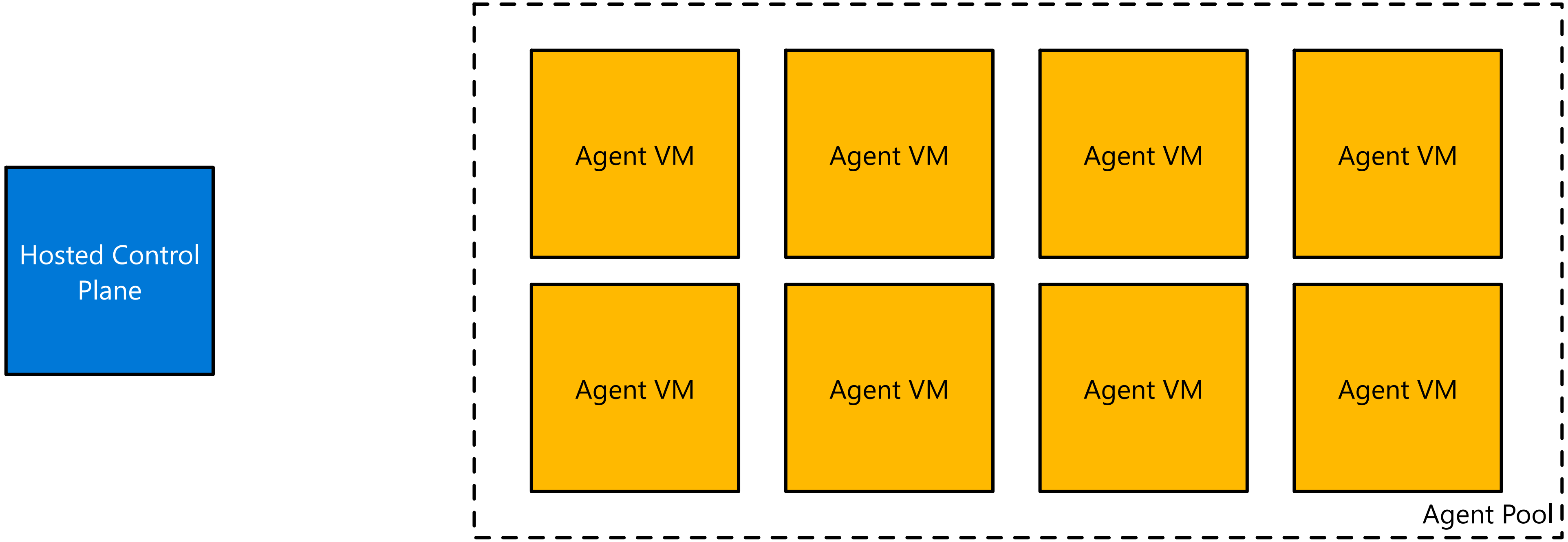
```
$ kubectl get nodes
NAME                       STATUS    AGE       VERSION
aks-mycluster-36851231-0   Ready     12m       v1.8.1
aks-mycluster-36851231-1   Ready     8m        v1.8.1
aks-mycluster-36851231-2   Ready     3m        v1.8.1
```

```
$ az aks scale -g myResourceGroup -n myCluster --agent-count 10
\ Running ..
```

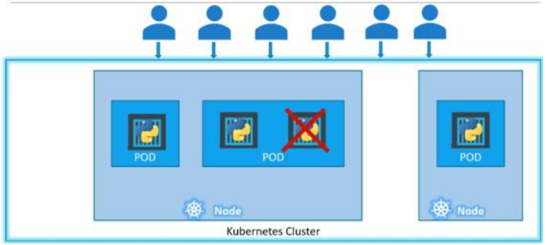
Kubernetes without AKS



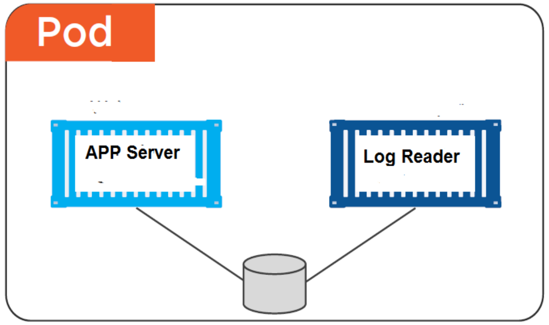
Kubernetes with AKS



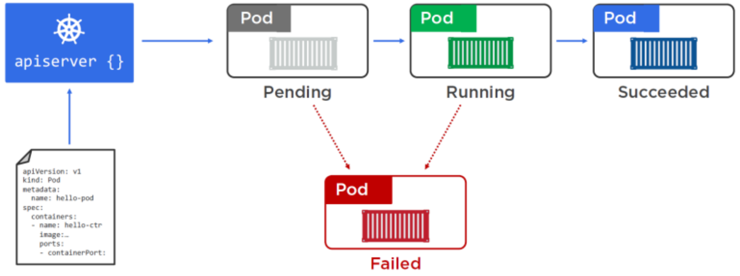
Cluster Components



Multi-container Pods



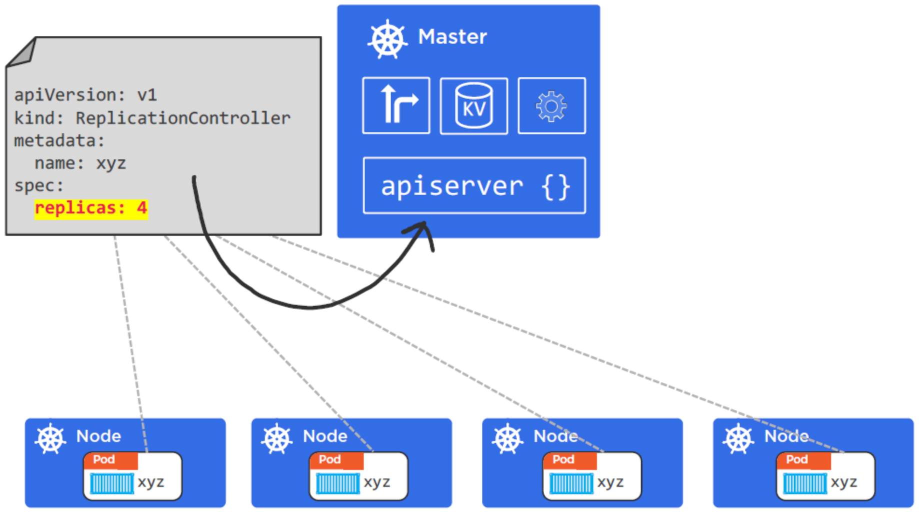
- **Pod**
- The most basic unit of work and Scheduling
- **A group of one or more containers which is your application or service**
- **The containers in pods can share storage, networking, and a specification on how to run the containers**
- The pod itself doesn't actually run anything. It's just a sandbox to run containers
- **Pod is never redeployed. If pod goes down a new pods is created .**
- **Pod Life cycle : creating , Pending, Running, succeeded**
- Kubernetes runs containers, but always inside of pods
- You can run more than one container inside of a pod,
- but it's a an advanced topic.



Cluster Components

kubernetes

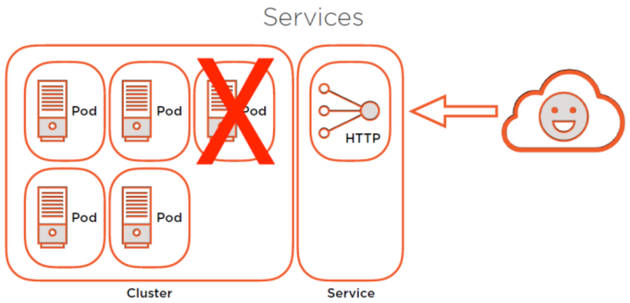
- **Deployment**
- A declarative method to define pods and replica sets usually in in a YAML file and we throw it at the apiserver .
- This is the method we will use to show how to define and configure HADR for SQL Server with k



```
apiVersion: v1
kind: ReplicationController
metadata:
  name: xyz
spec:
  replicas: 4
```

Cluster Components

- **Service**
- Networking abstraction : A logical set of pods that can be abstracted.
- One type of service is a load balancer for connectivity. You need to configure IP and DNS name for the service. Each pod has a unique IP address, but a load balancer is a known IP address. Like the virtual IP address concept used by Failover Cluster Instance or the listener for Availability Groups.
- Scaled by adding/removing Pods



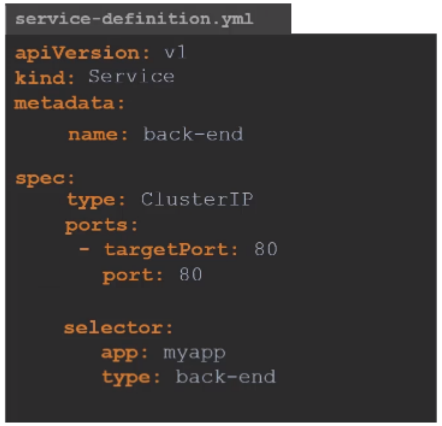
Cluster Components

kubernetes

-
- **Persistent volume claim**
- Storage that can be used by pods through a PersistentVolume. This storage can be shared across pods **(as a shared storage HADR solution for SQL Server).**
- A Persistent volume claim is a request by a user for a PersistentVolume storage.
-
  -

YAML File

-
- **What is YAML ?**
- Markup Language
- Easy to read
- **Two Kubernetes Object fields**
- Object Spec
- Object status
-

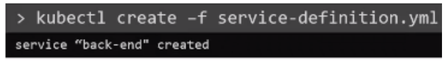


service-definition.yml

```
apiVersion: v1
kind: Service
metadata:
    name: back-end

spec:
    type: ClusterIP
    ports:
     - targetPort: 80
       port: 80

    selector:
       app: myapp
       type: back-end
```

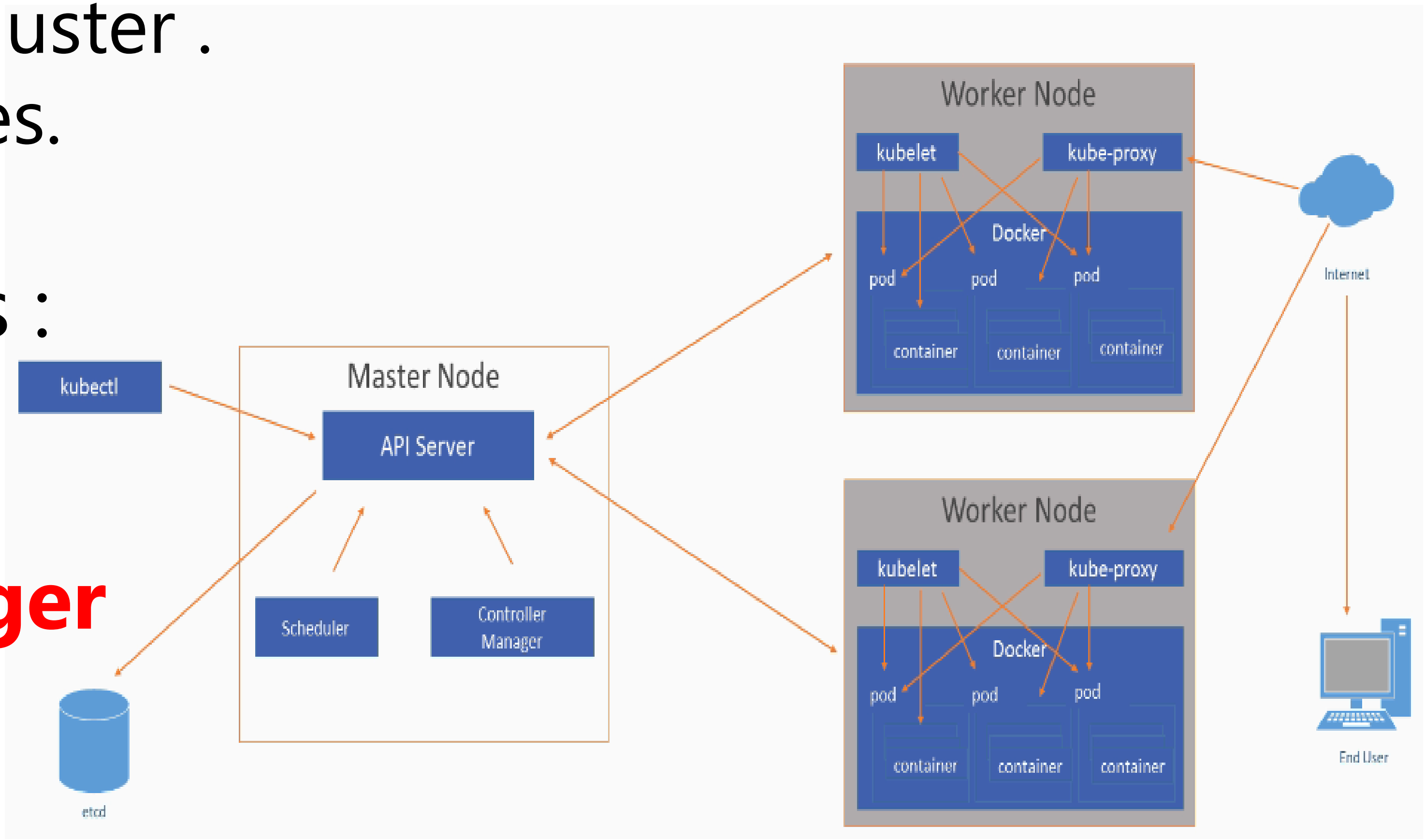


```
> kubectl create -f service-definition.yml
service “back-end” created
```

Kubernetes Architecture

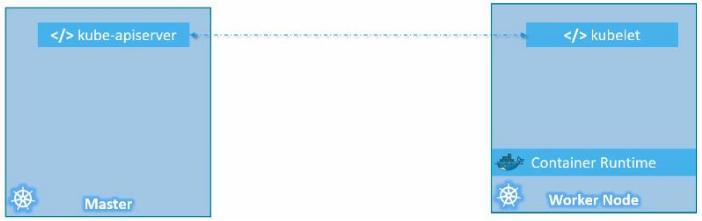
- **master node**
- Manages the K8S Cluster .
- Watches other nodes.

- It has 3 components :
- **API Server**
- **Scheduler**
- **Controller manager**
-

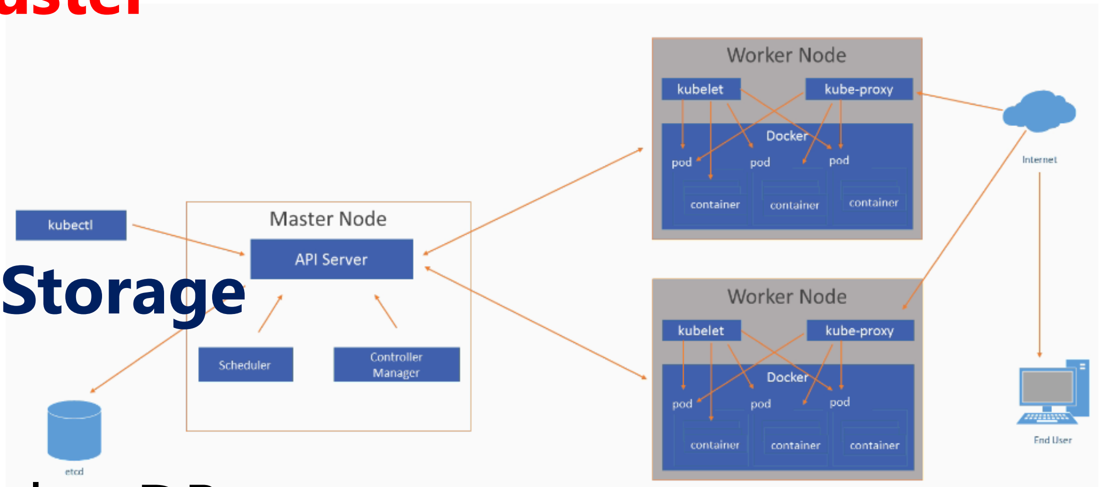


Kubernetes Architecture

•API Server

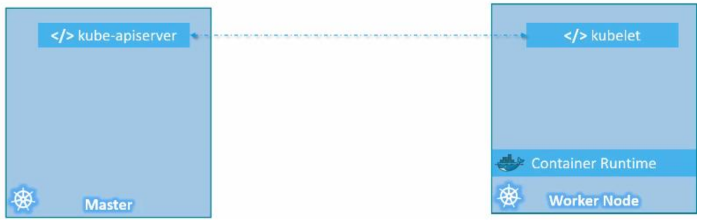


- **The front End**
- **The only way to user to interact with the cluster**
- **The Only way K8s interacts with the cluster**
- **RESTful API over HTTP using JSON**
- API Objects includes : **Nodes , Pods ,**
- **Replica Sets , Controllers, Services ,Storage**
- **and more**
- Serialized and persisted with **etcd** key value DB

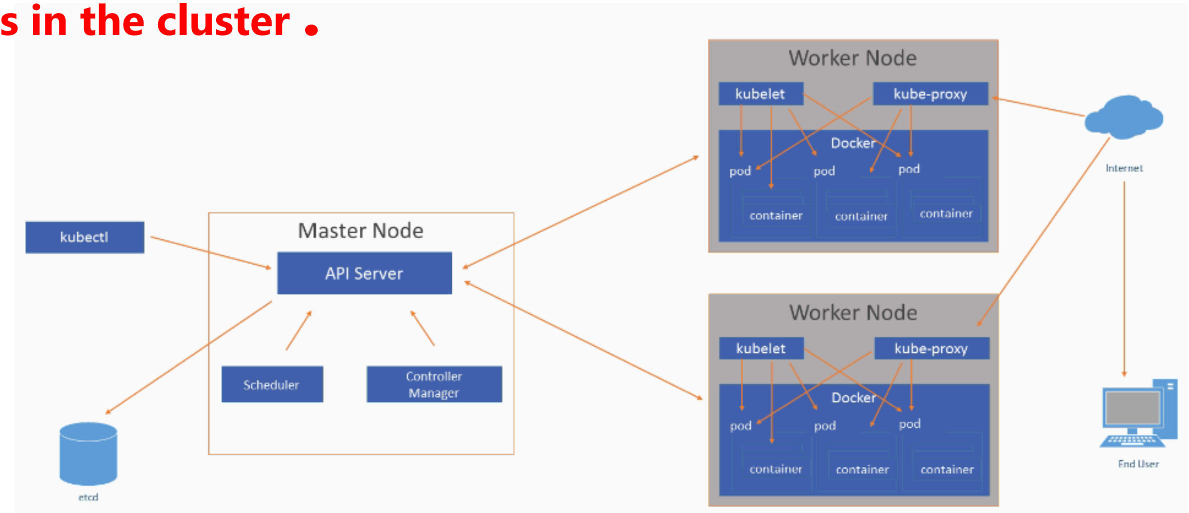


Kubernetes Architecture

- **Scheduler**
- **Watches apiserver for new pod**
- **Assigns work to nodes**
- **Manages the pods to run on which nodes.**



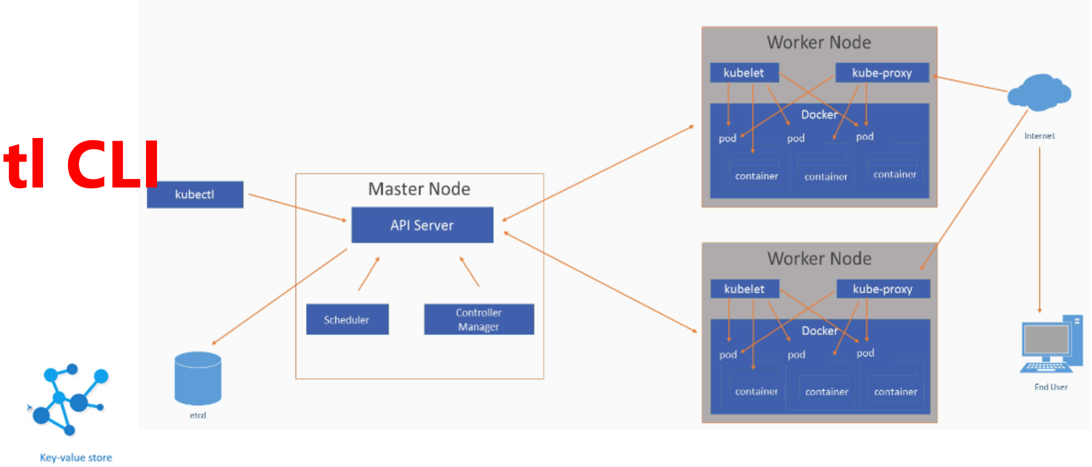
- **Controller manager**
- **Controller of controllers :background threads that run tasks in the cluster .**
- responsible for orchestration
- Watches for changes
- Helps maintain desired state
- Replication controllers which maintaining the
- replication number of pods.
- ReplicaSet : Number of replicas
- Deployment: Manage rollout of ReplicaSet
- Endpoint controllers : Join services and ports together

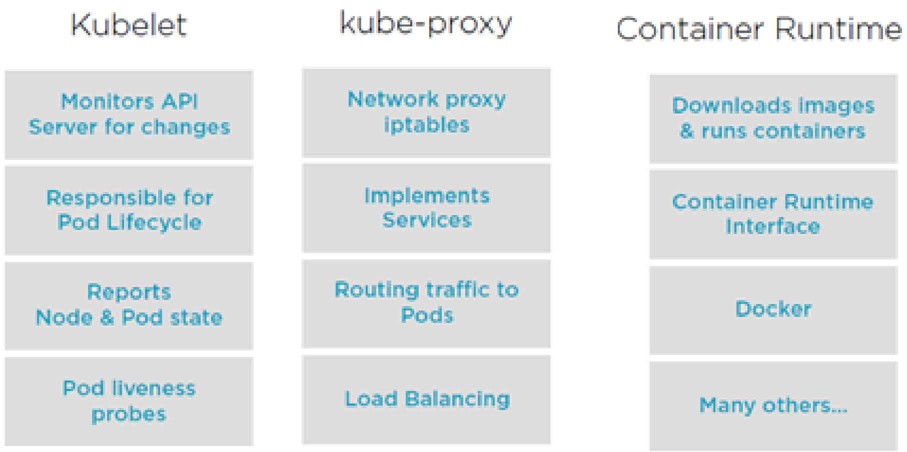


Kubernetes Architecture

- **kubectl**
- **Interact with master node using kubectl CLI**
- **Worker nodes**
- **Worker nodes which can be VMs or**
- **Physical servers .**
- **Kubelet Agent**
- **The communication between nodes**
- **and master node is done with kubelet .**
- kubelet is agent which interact which API Serve to see if the pods have been assigned to the node ,
- it also deploy container on the worker nodes .
- Container node should have Docker platform .
- **Kube-proxy**
- **Network proxy and load balancer for network**

kubernetes



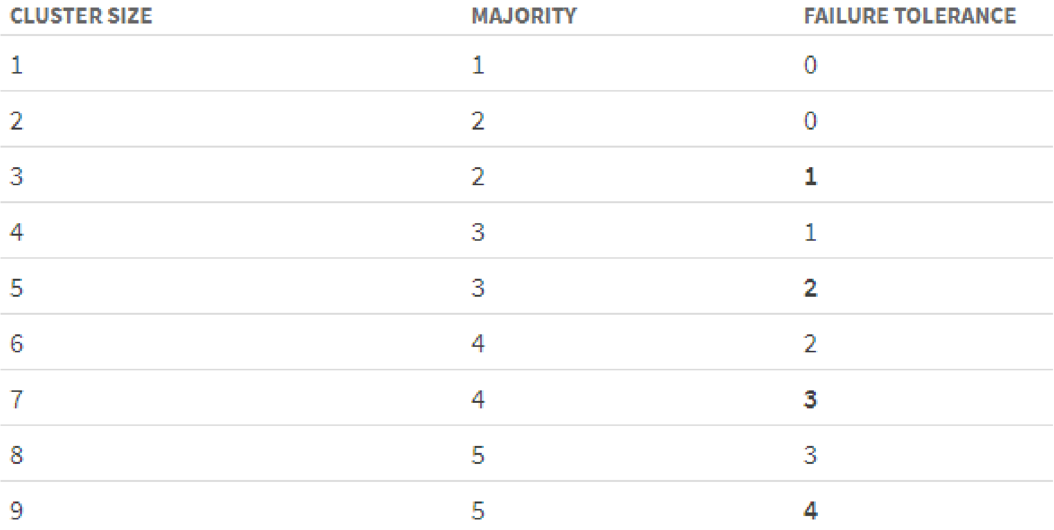


Kubelet	kube-proxy	Container Runtime
Monitors API Server for changes	Network proxy iptables	Downloads images & runs containers
Responsible for Pod Lifecycle	Implements Services	Container Runtime Interface
Reports Node & Pod state	Routing traffic to Pods	Docker
Pod liveness probes	Load Balancing	Many others...

Kubernetes Architecture

- **etcd**

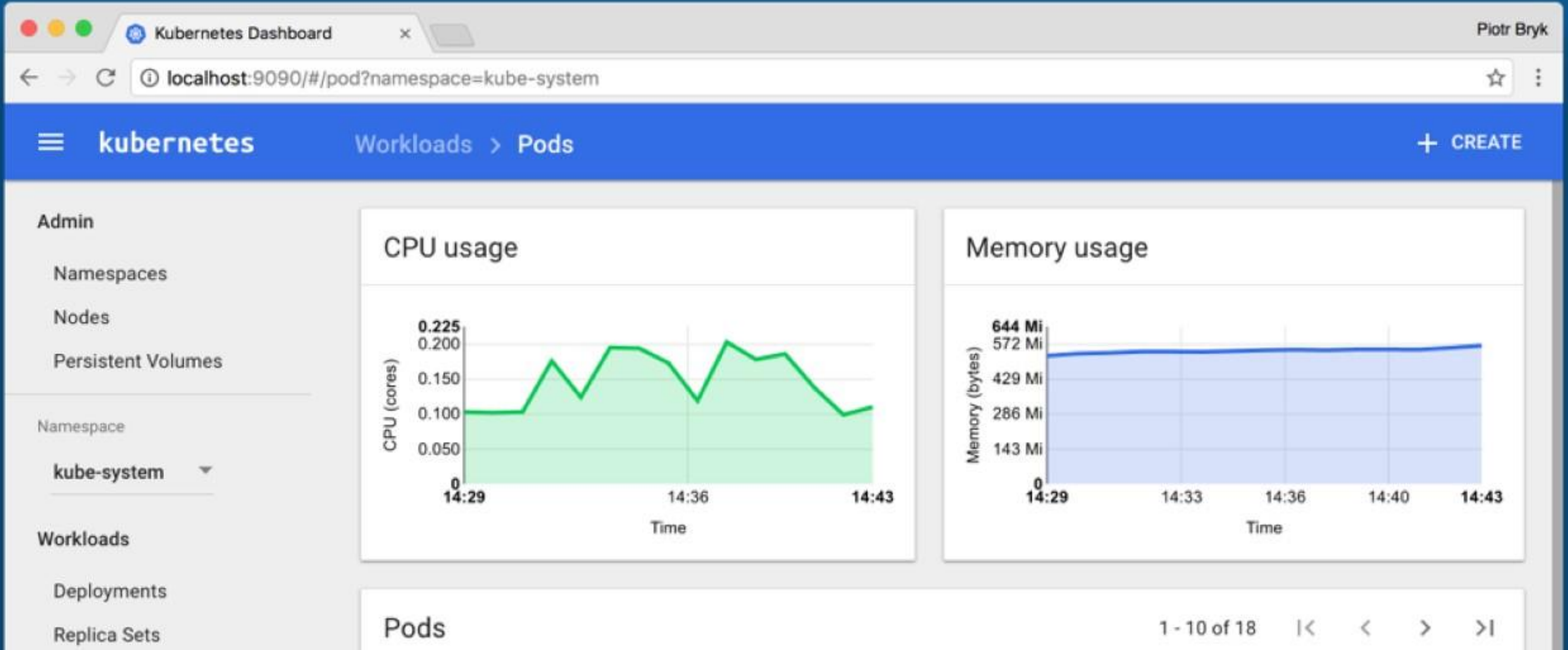
- A light weight highly resilient key value store DB that originates from CoreOs.
- k8s Store all Cluster state and config .
- The *source of truth* .
- Have a backup plan for it!
- The CoreOs administration guide document states that a minimum of 3 etcd members (instances) are required in order for the cluster to tolerate failures, as quoted verbatim from the documentation vis:



CLUSTER SIZE	MAJORITY	FAILURE TOLERANCE
1	1	0
2	2	0
3	2	**1**
4	3	1
5	3	**2**
6	4	2
7	4	**3**
8	5	3
9	5	**4**

Access the Dashboard

```
az aks browse --resource-group myResourceGroup --name
myAKSCluster
```



localhost:9090/#/pod?namespace=kube-system

Delete an AKS Cluster

```
az aks delete --resource-group myResourceGroup --name
myAKSCluster
```

Argument	Description	Required
--name	Resource name for the managed cluster	Yes
--resource-group	Name of the AKS resource group	Yes
--no-wait	Do not wait for the long-running operation to finish	No
--yes	Do not prompt for confirmation	No

Azure Container Service (AKS)

Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Release automation tools

Simplifying the Kubernetes experience

DRAFT

Streamlined Kubernetes development

HELM

The package manager for Kubernetes

BRIGADE

Event-driven scripting for Kubernetes

KASHTI

Visualization dashboard for Brigade

Azure Container Service (AKS)

Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Helm

The best way to find, share, and use software built for Kubernetes

Manage complexity

Charts can describe complex apps; provide repeatable app installs, and serve as a single point of authority

Easy updates

Take the pain out of updates with in-place upgrades and custom hooks

Simple sharing

Charts are easy to version, share, and host on public or private servers

Rollbacks

Use `helm rollback` to roll back to an older version of a release with ease

Azure Container Service (AKS)

Azure Container Instances (ACI)

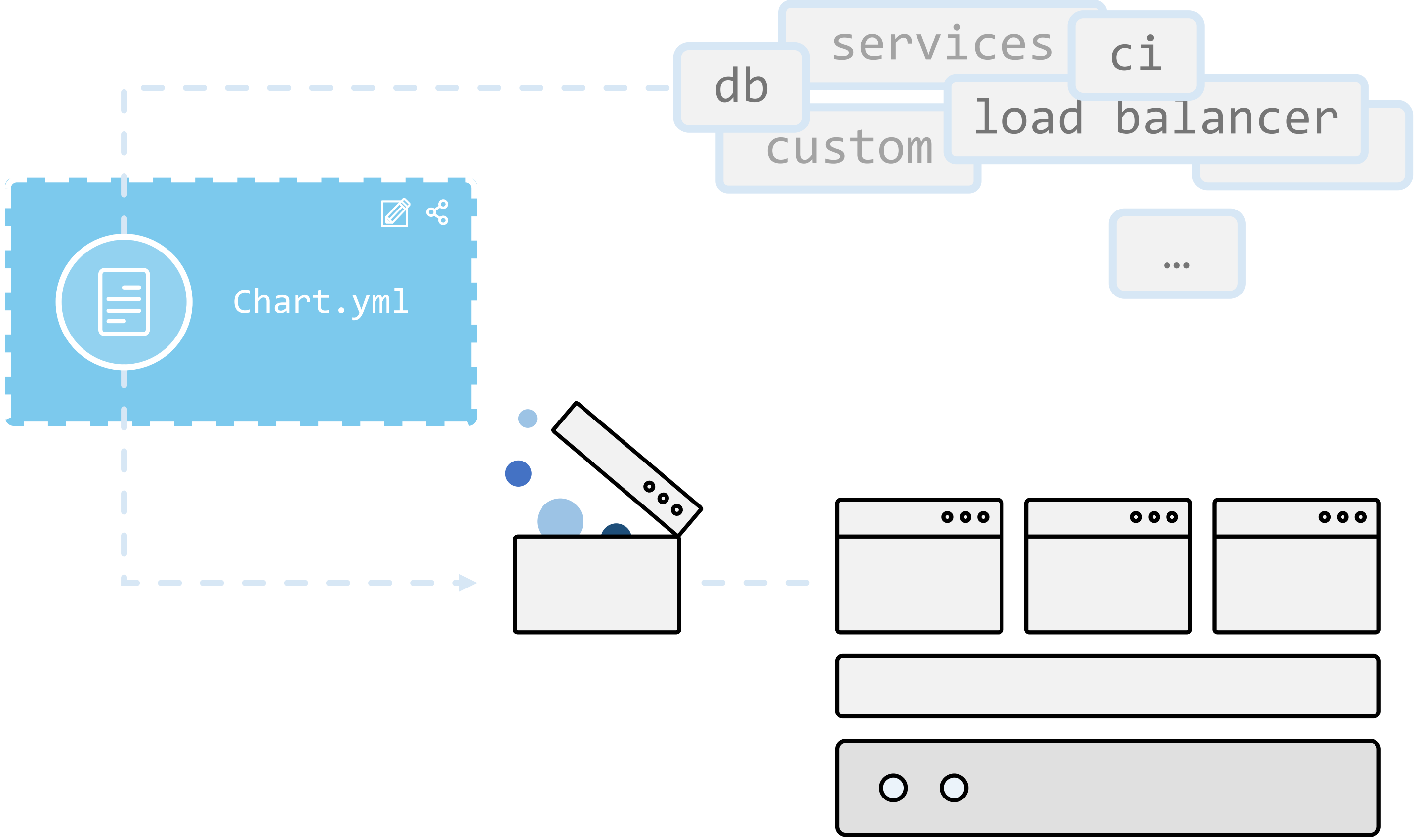
Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Helm

Helm Charts helps you define, install, and upgrade even the most complex Kubernetes application



Installing Helm

Helm.sh

Installing Draft

draft.sh

Installing Draft

draft init to set up draft (after prerequisites are installed)
draft create to containerize your application based on Draft packs
draft up to deploy your application to a Kubernetes dev sandbox, accessible using draft connect over a secured tunnel.

Use a local editor to modify the application, with changes deployed to Kubernetes in seconds.

Installing Brigade

brigade.sh

Brigade

- Brigade is a tool for running scriptable, automated tasks in the cloud — as part of your Kubernetes cluster.

```
# add Brigade chart repo
helm repo add brigade https://brigadecore.github.io/charts

# install Brigade
helm install -n brigade brigade/brigade

# if you want to activate Generic Gateway, you should use this command
# helm install -n brigade brigade/brigade --set genericGateway.enabled=true
```

Installing Kashti

https://github.com/brigadecore/kashti