THANK YOU TO OUR SPONSORS

Presented by Microsoft

Building and Deploying Containers using Docker and Azure

VAIBHAV GUJRAL
MICROSOFT MVP - AZURE

About Me

- Cloud Architect at Kiewit Corp
- Microsoft Most Valuable Professional (MVP) – Azure
- Microsoft Certified Trainer (MCT)
- Microsoft Certified Azure Solutions Architect Expert
- Organizer, Omaha Azure User Group
- http://www.vaibhavgujral.com
- @vabgujral
- linkedin.com/in/vaibhavgujral/
- #AzureHeroes Community & Content Hero
- Listed among the “Top 50 Microsoft Azure Blogs, Websites & Influencers in 2020”

Agenda

What are containers?
- Containers vs Virtual Machines

Docker basics

Azure Container Registry

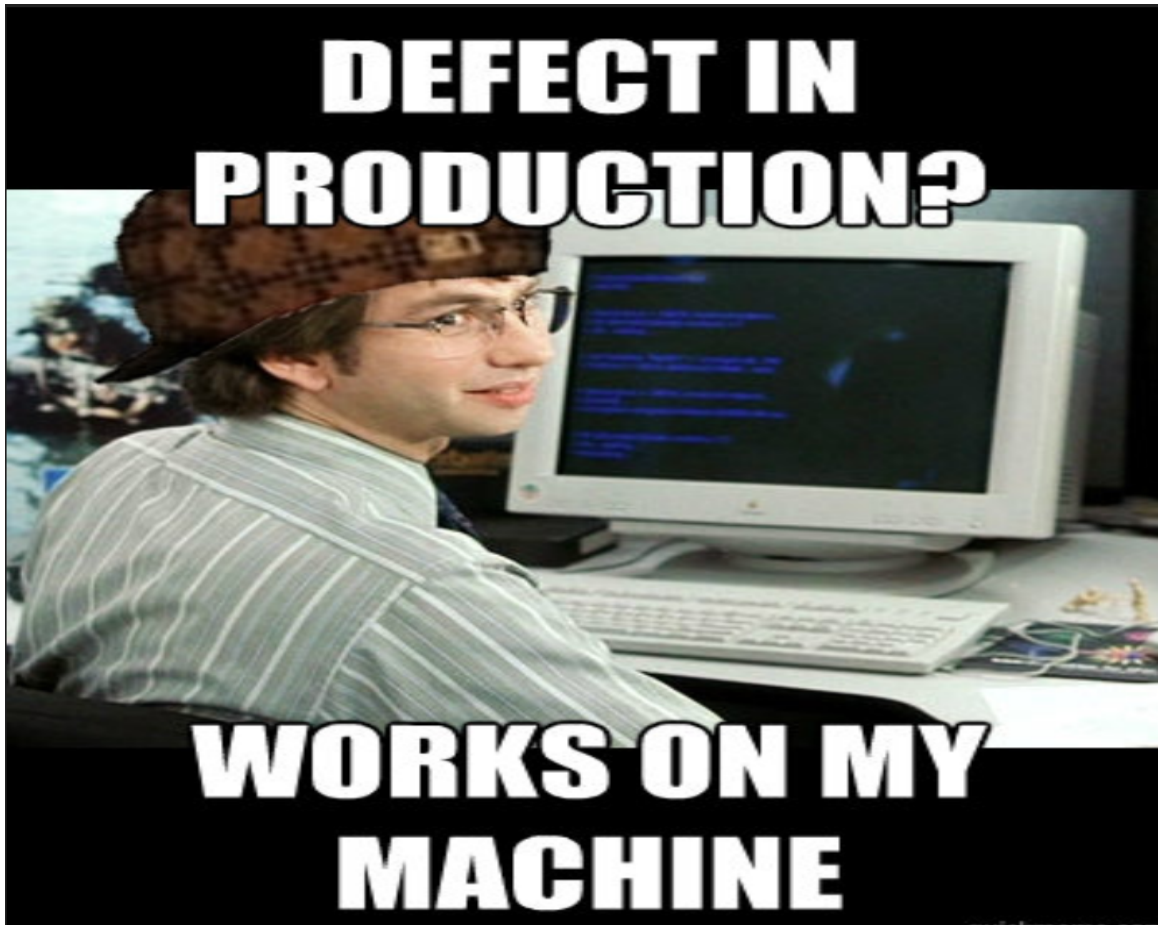
Azure Container Instances

Azure Web App for Containers

Azure Kubernetes Services

Q&A

Can you relate to this?

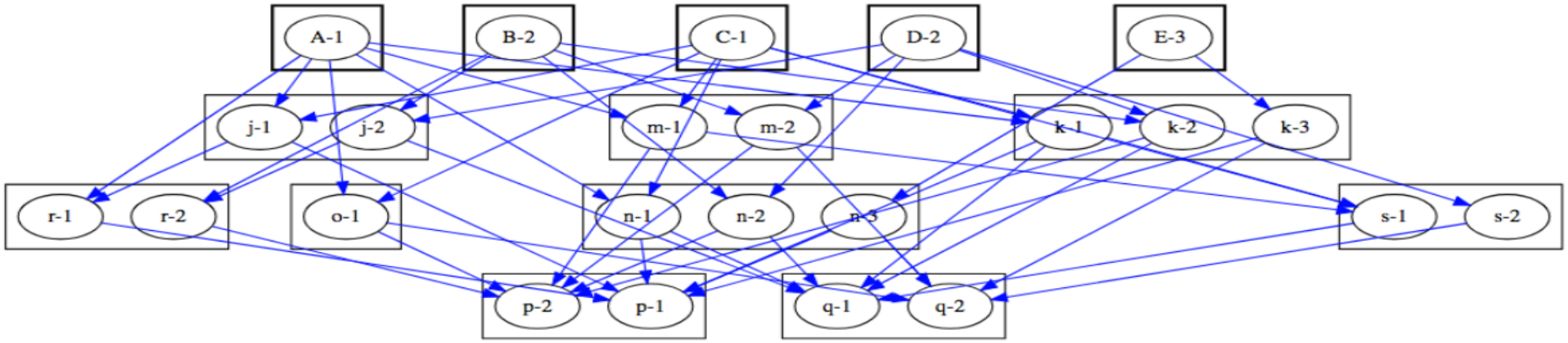


“Dependency hell”

Dependency hell

From Wikipedia, the free encyclopedia

Dependency hell is a colloquial term for the frustration of some software users who have installed software packages which have dependencies on specific versions of other software packages.[1]



Dependency Hell

Solutions to “Dependency hell”

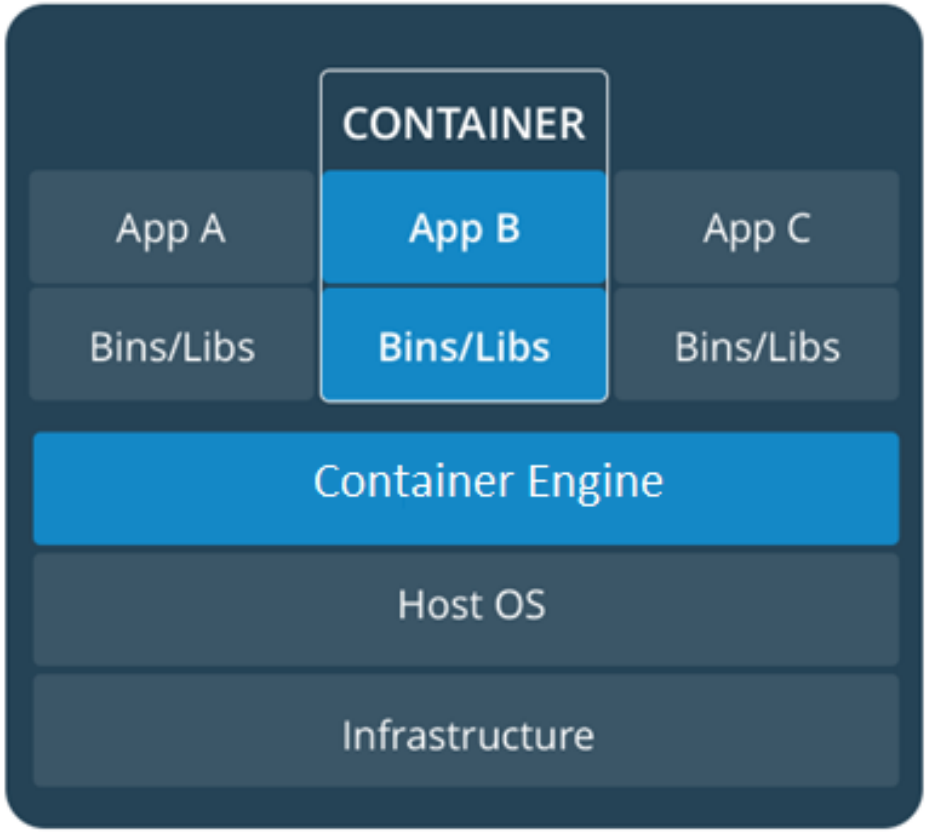
1. Private per-application versions
2. Side-by-side installations
3. Smart package management
4. Software appliances
5. Installer options
6. Portable self-contained applications

What are containers?

Containers provide a standard way to package application's code, configurations, and dependencies into a single unit of deployment

Containers share an operating system installed on the server and run as resource-isolated processes

Containers ensure quick, reliable, and consistent deployments, regardless of environment – Linux, Windows, Data center, Serverless…



Containers Terminology

An **image** is an executable package that includes everything needed to run an application--the code, runtimes, libraries, environment variables, and configuration files.

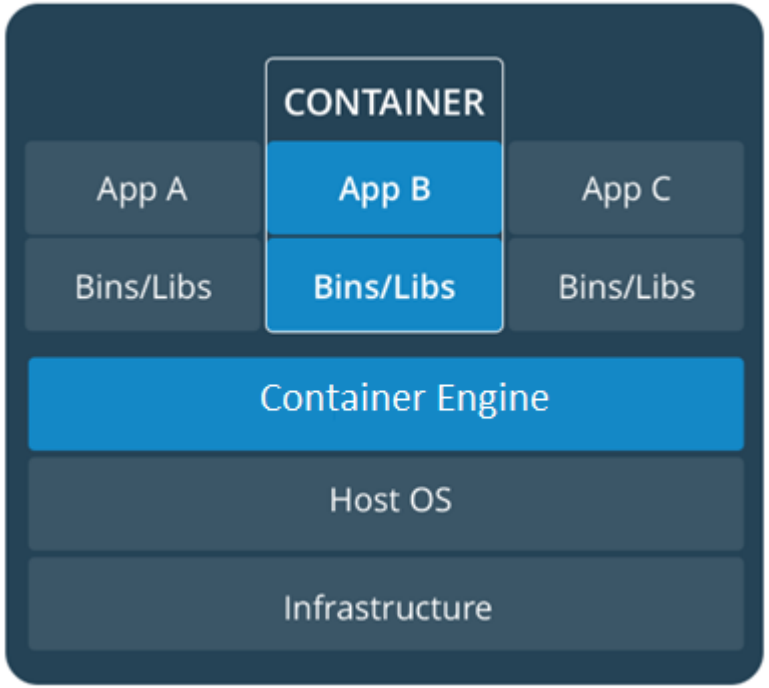
A **container** is a runtime instance of an image--what the image becomes in memory when executed (that is, an image with state, or a user process).

A **container registry** is a storage for container images that allows image distribution. Container registry can be private or public registry

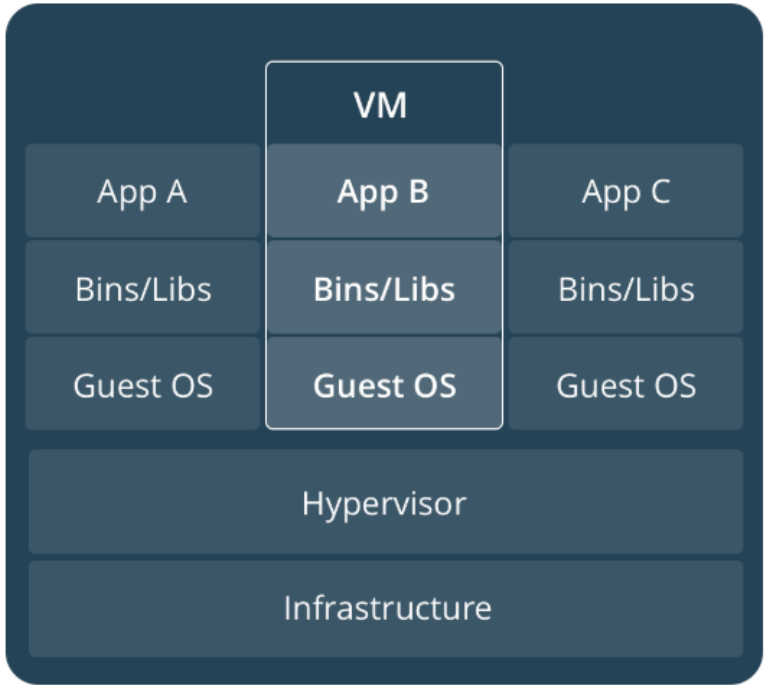
Some of the widely used container registries are DockerHub, Azure container registry, AWS ECR (EC2 Container Registry), and Google Container Registry

Containers vs virtual machines

CONTAINERS



VIRTUAL MACHINES



Containers vs virtual machines

CONTAINERS

- Abstraction at app level
- Each container runs as an isolated process in the user space
- Multiple containers share the same OS kernel
- Lightweight (few MBs)
- Portable and efficient
- Faster to build and run
- Less Isolated than VMs

VIRTUAL MACHINES

- Abstraction at OS level
- Hypervisor enables running multiple VMs on a single server
- Each VM has its own copy of operating system
- Heavyweight (GBs) and wastage of resources
- Portable
- Slow to boot

What is Docker?

Docker is an open source project for automating the deployments of applications as portable self-sufficient containers that can run on cloud or on-premises

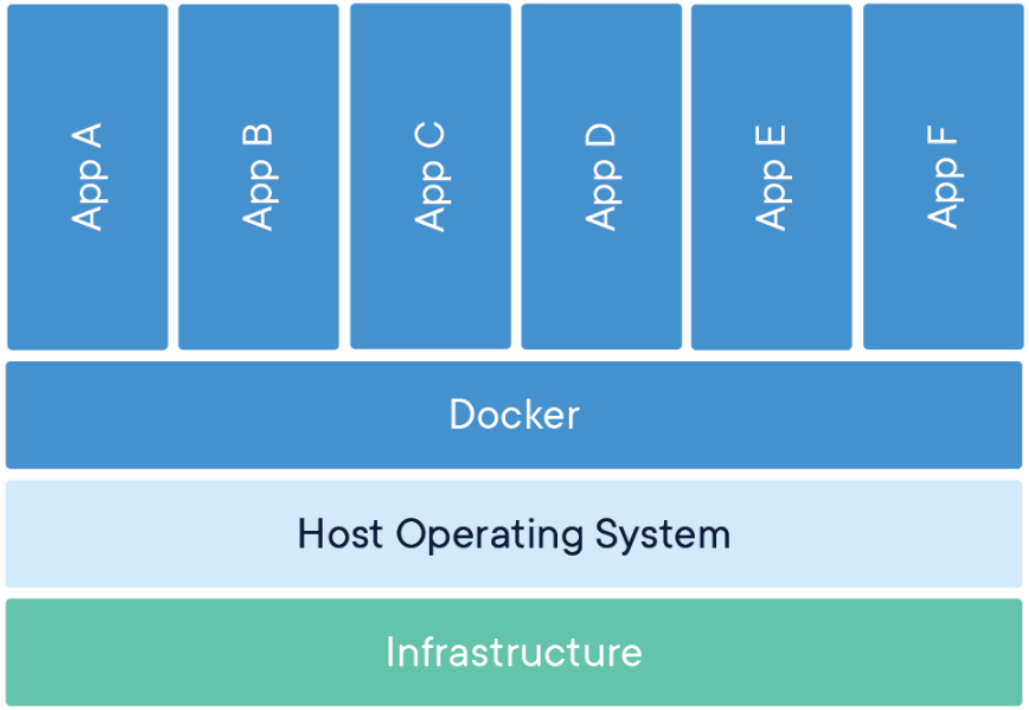
Docker is also the name of the company which promotes and evolves the Docker project

Docker Engine, a software daemon, is the core of Docker

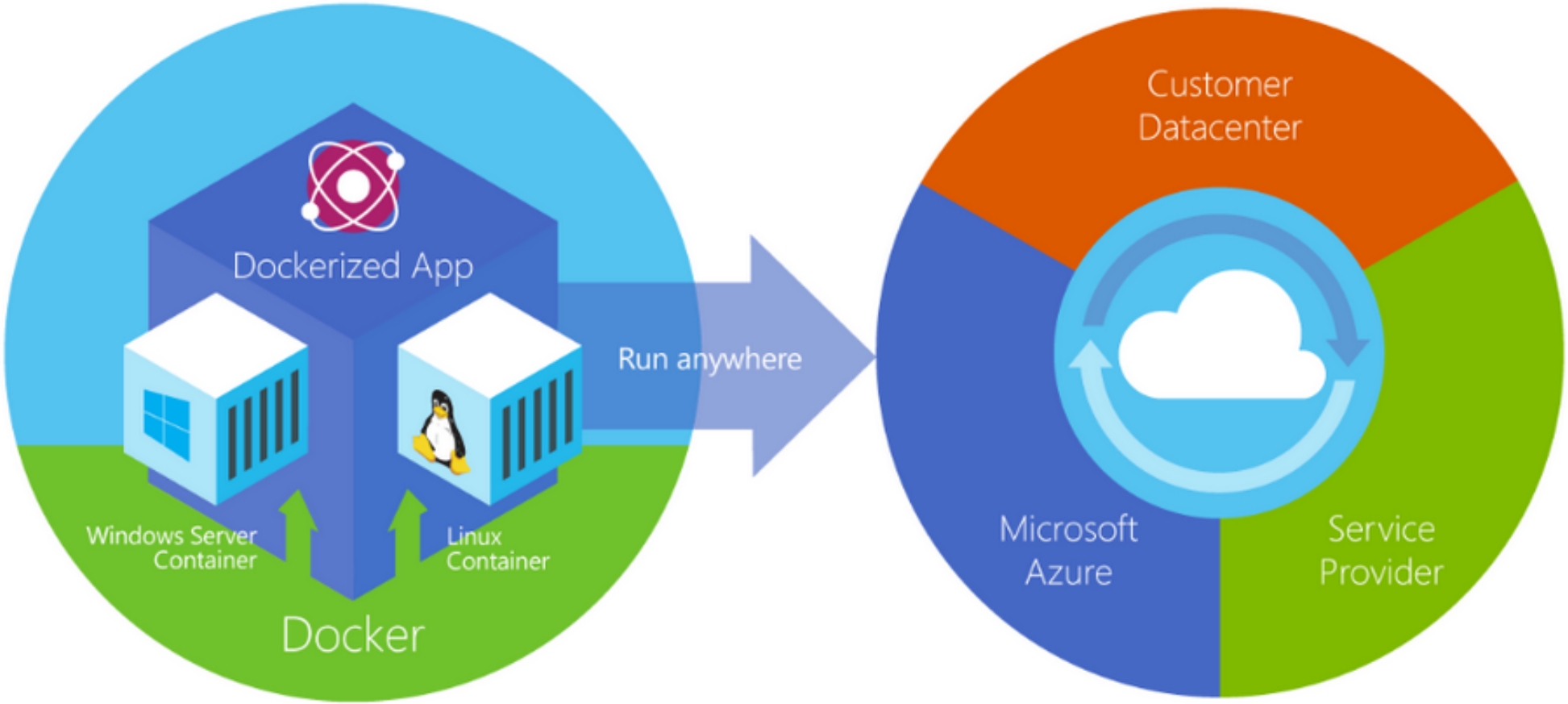
Docker Containers run on top of Docker Engine

Docker Hub is the most widely used container registry

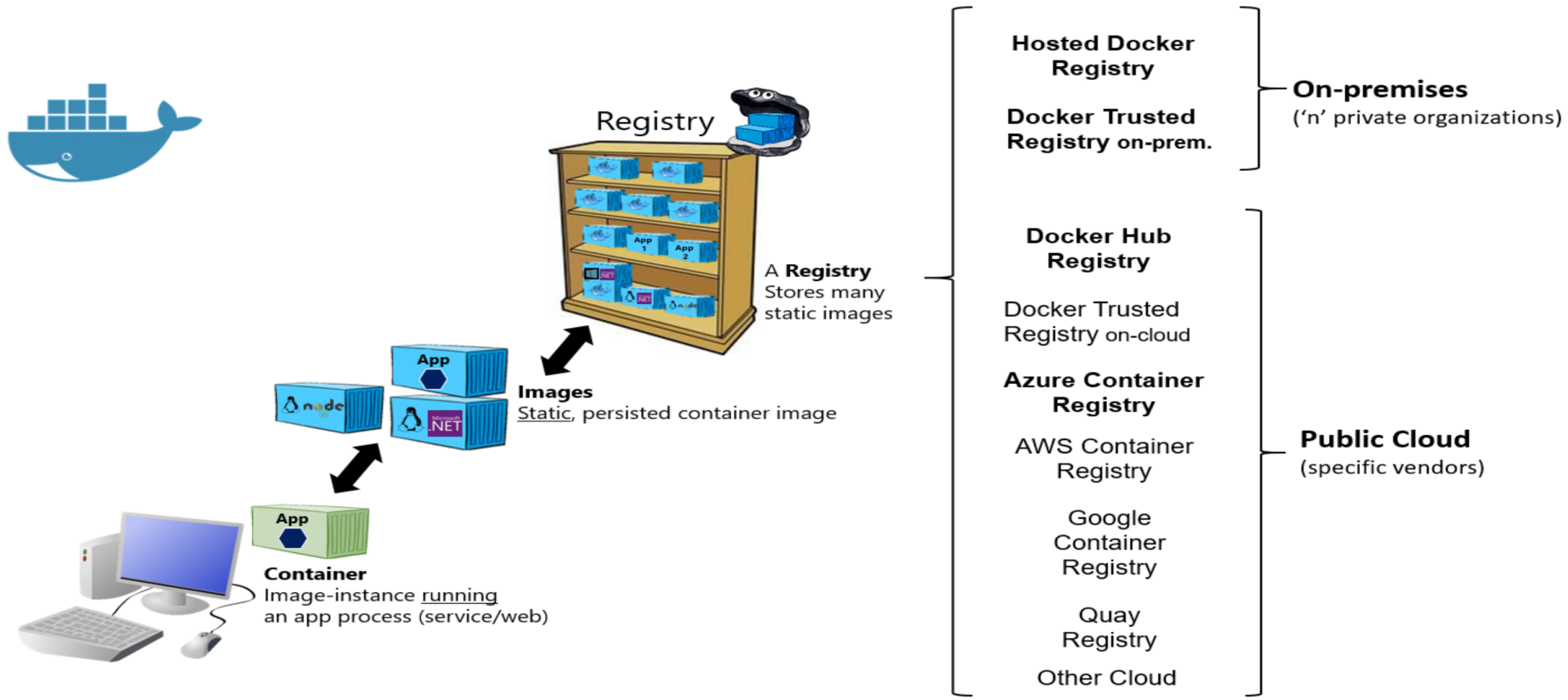
Docker Trusted Registry is the enterprise-grade image storage solution from Docker which can be installed on-premises



Build Once, run anywhere



Basic Taxonomy in Docker



Installing Docker

Two editions –

1. Docker Community Edition – for development environments
2. Docker Enterprise Edition – for enterprise development

Docker Community Edition also known as Docker Desktop

Docker Desktop includes Docker Engine, Docker CLI client, Docker Compose, Docker Machine and Kitematic

Switch between Windows and Linux Containers in Windows

https://www.docker.com/products/docker-desktop

Docker CLI

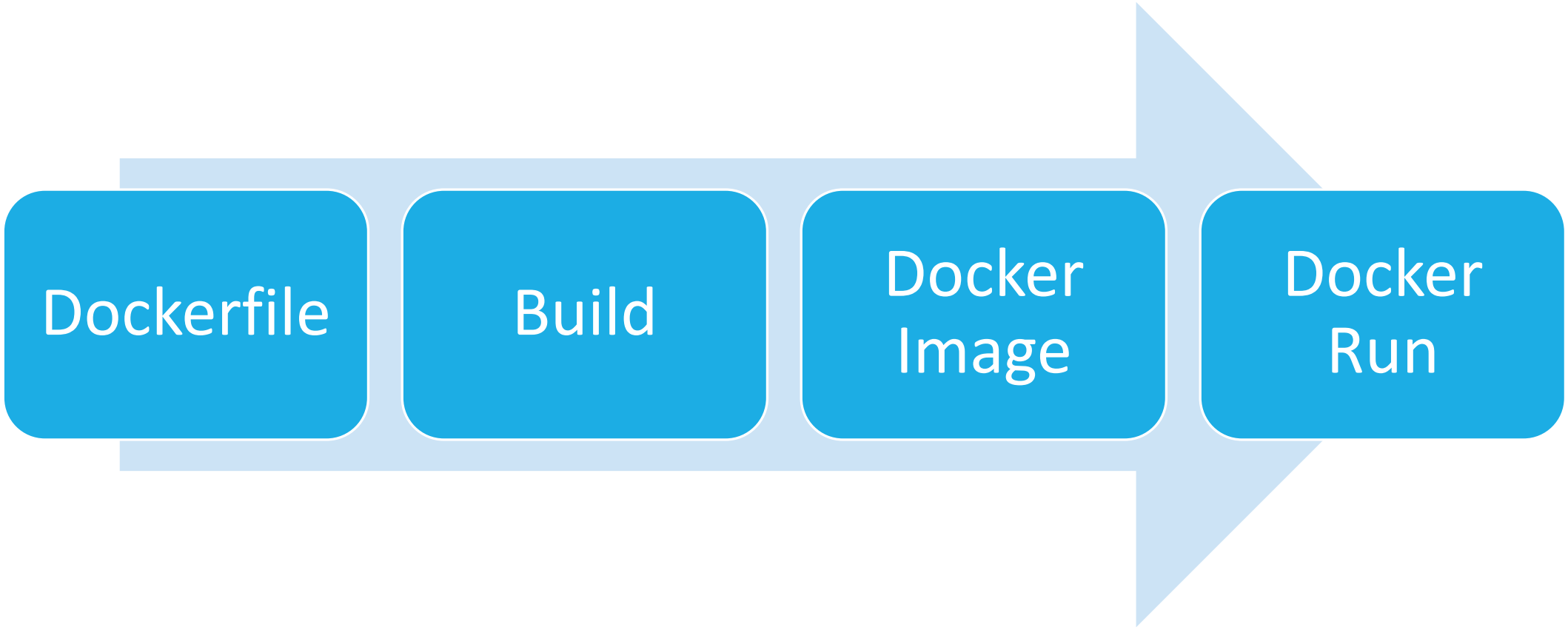
```
$ docker
Usage: docker [OPTIONS] COMMAND [ARG...]
       docker [ --help | -v | --version ]

A self-sufficient runtime for containers.
```

To list available commands, either run `docker` with no parameters or execute `docker help`

Run `docker --version` and ensure that you have a supported version of Docker:

Run `docker info` (or `docker version` without `--`) to view even more details about your Docker installation

Docker Build Process



Dockerfile

Dockerfile is a text file containing instructions for how to build docker images

It includes all the instructions needed by Docker to build the image.

A Docker image is made up of a series of filesystem layers representing instructions in the image's Dockerfile that makes up an executable software application.

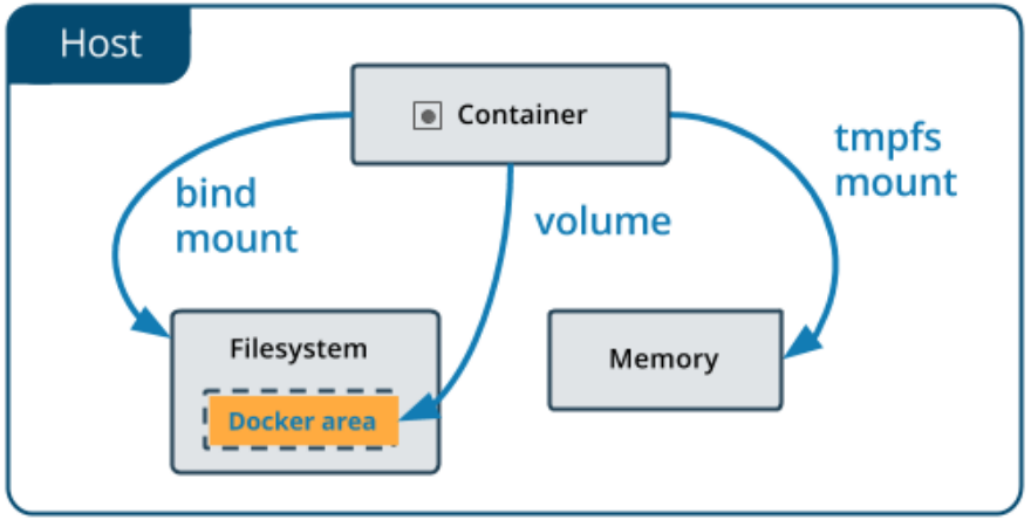
Docker build action builds a container image based on the information provided in the dockerfile

```
FROM mcr.microsoft.com/dotnet/core/sdk:3.0 AS build
WORKDIR /app

# copy csproj and restore as distinct layers
COPY *.sln .
COPY aspnetapp/*.csproj ./aspnetapp/
RUN dotnet restore

# copy everything else and build app
COPY aspnetapp/. ./aspnetapp/
WORKDIR /app/aspnetapp
RUN dotnet publish -c Release -o out


FROM mcr.microsoft.com/dotnet/core/aspnet:3.0 AS runtime
WORKDIR /app
COPY --from=build /app/aspnetapp/out ./
ENTRYPOINT ["dotnet", "aspnetapp.dll"]
```

https://docs.docker.com/engine/reference/builder/

Storage options

1. **Volumes** are stored in a part of the host filesystem which is managed by Docker. Non-Docker processes should not modify this part of the filesystem. Volumes are the best way to persist data in Docker. Volumes are managed by Docker and are isolated from the core functionality of the host machine.
2. **Bind mounts** may be stored anywhere on the host system. They may even be important system files or directories. Non-Docker processes on the Docker host or a Docker container can modify them at any time.
3. **tmpfs mounts** (only in Linux) are stored in the host system's memory only and are never written to the host system's filesystem.



Docker Compose

Docker Compose is a command-line tool and YAML file format with metadata for defining and running multi-container applications.

Allows defining a single application based on multiple images with one or more .yml files that can override values depending on the environment.

After the definitions are created, the entire multi-container application can be deployed by using a single command (**docker-compose up**) that creates a container per image on the Docker host.

Using Compose is a three-step process:

1. Define application's environment with a Dockerfile so it can be reproduced anywhere
2. Define the services that make up the application in docker-compose.yml so they can be run together in an isolated environment
3. Run docker-compose up and Compose starts and runs entire application

Demo

RUNNING DOCKER CONTAINERS LOCALLY

Azure Container registry

First-class Azure resource

Managed, private Docker registry service based on the open-source Docker Registry 2.0.

Can be used with existing container development and deployment pipelines

Use Azure Container Registry Tasks to build container images in Azure

Three pricing tiers:

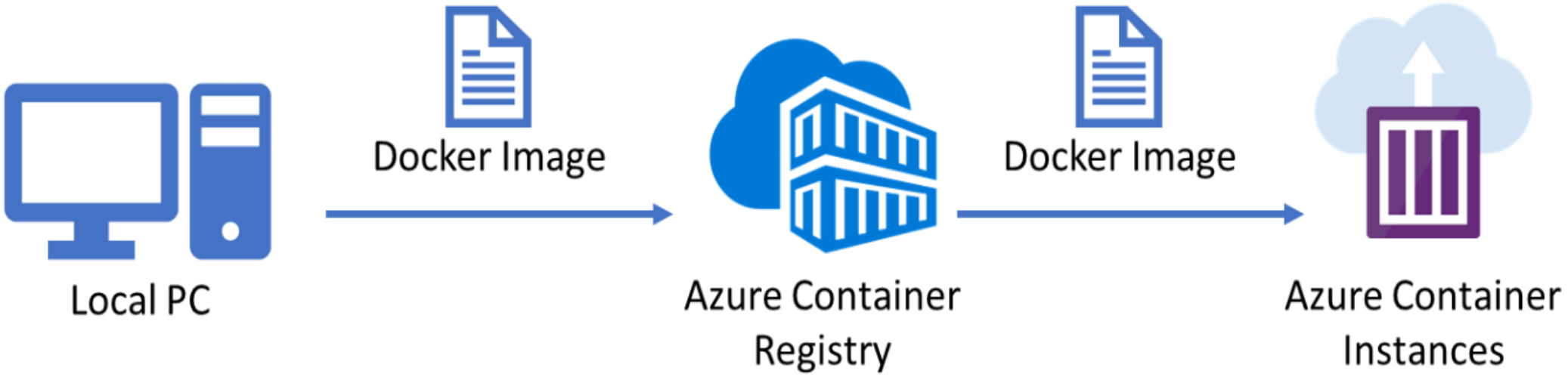
1. Basic
2. Standard
3. Premium

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

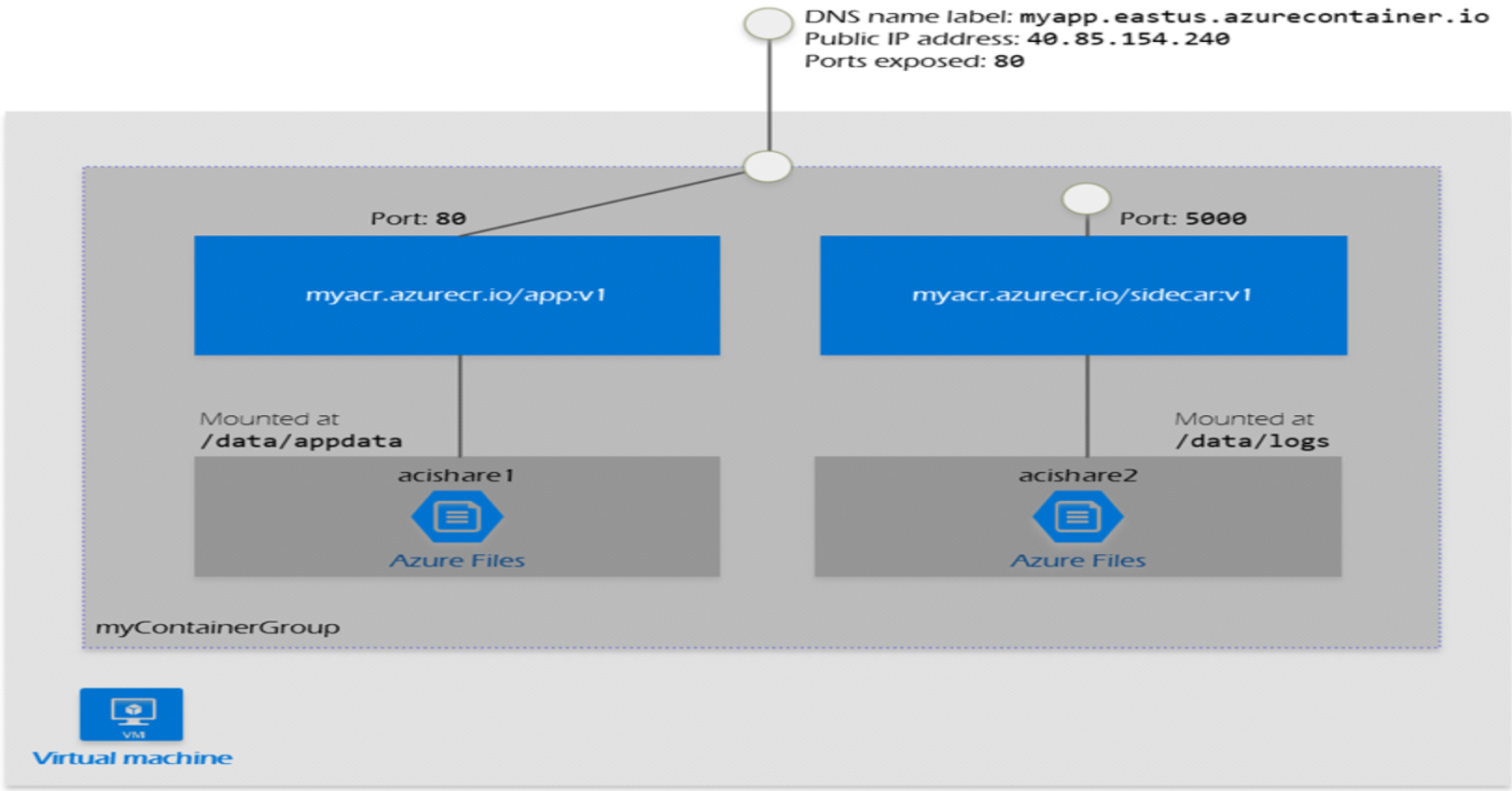
Azure container instances

Fastest and simplest way to run a container in Azure

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

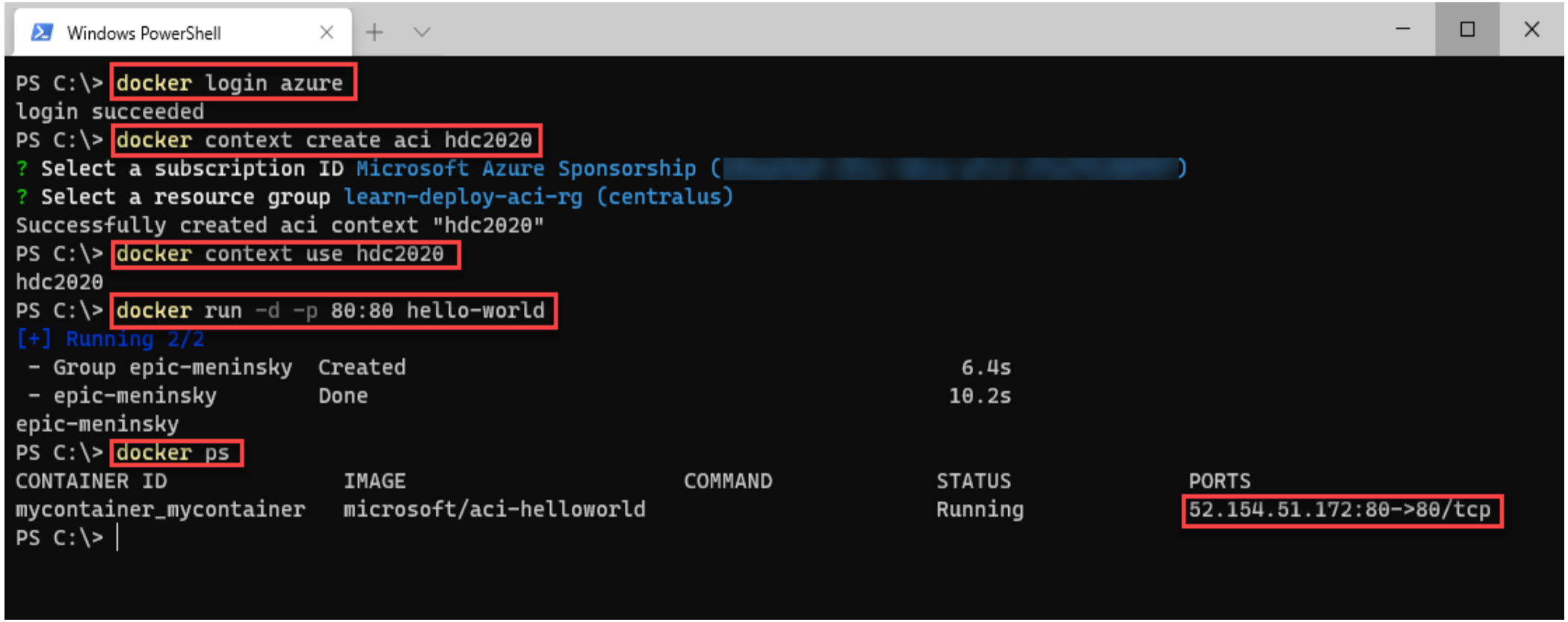


Container Groups



https://docs.microsoft.com/en-us/azure/container-instances/container-instances-container-groups

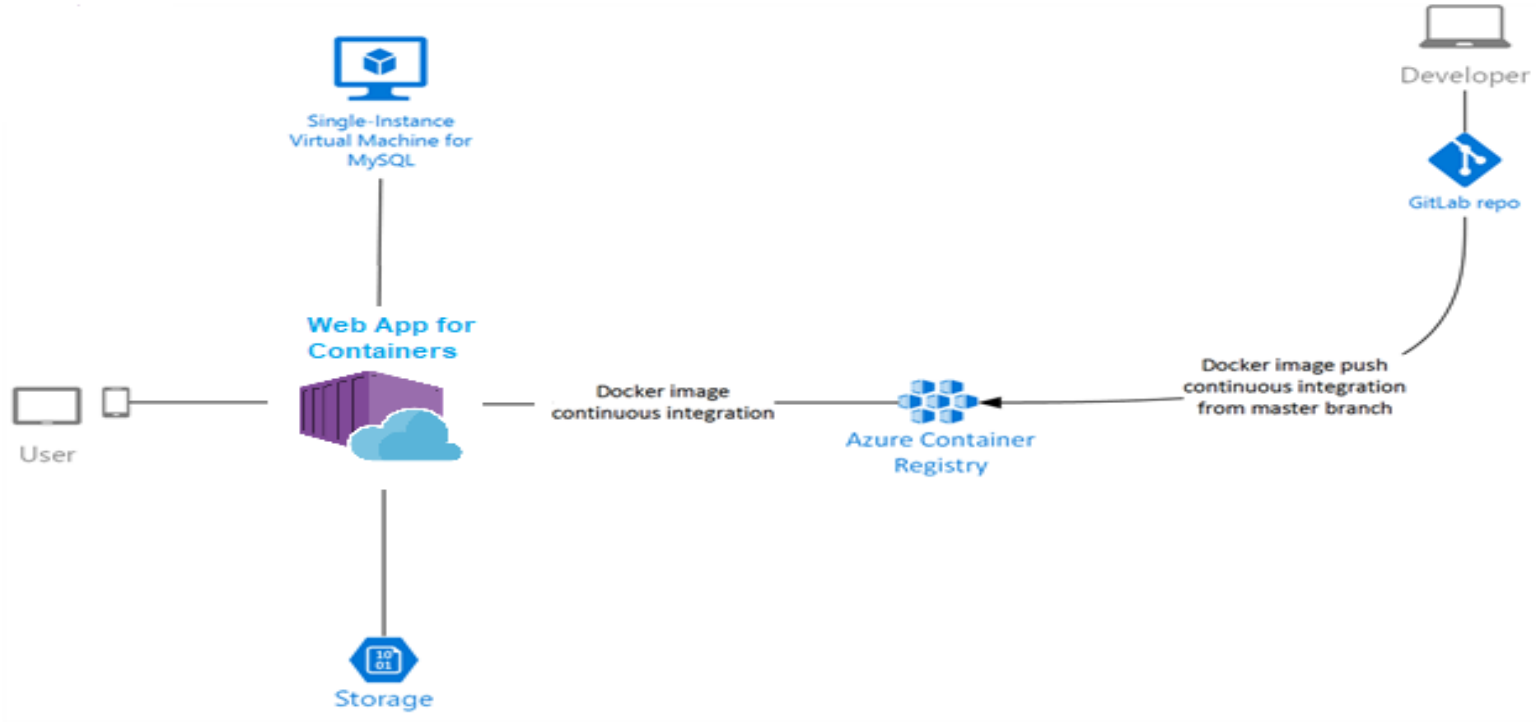
ACI Support in Docker CLI



https://docs.docker.com/engine/context/aci-integration/

Azure web apps for containers

Easily deploy and run containerized applications on Windows and Linux



https://azure.microsoft.com/en-us/services/app-service/containers/

Demo

RUNNING A CONTAINER IN AZURE CONTAINER INSTANCES

RUNNING A CONTAINER IN AZURE WEB APPS FOR CONTAINERS

Container orchestration

Enables scaling out applications across many docker hosts

Group of these hosts known as cluster

Orchestrators abstracts the complexities of the underlying platform by managing multiple hosts as a single cluster

Example –

1. Docker Swarm
2. Kubernetes
3. Azure Service Fabric
4. Mesosphere DC/OS

What is Kubernetes?

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.

Kubernetes provides a container-centric infrastructure that groups application containers into logical units for easy management and discovery.

Kubernetes supports automatic deployment, scaling, and operations of application containers across clusters of hosts

Kubernetes provides features including Service Discovery & load balancing, Service Orchestration, automated rollouts and rollbacks, self-healing and secret & configuration management

https://kubernetes.io/

Azure Kubernetes service

Hosted Kubernetes service in Azure

Reduces the complexity and operational overhead of managing Kubernetes

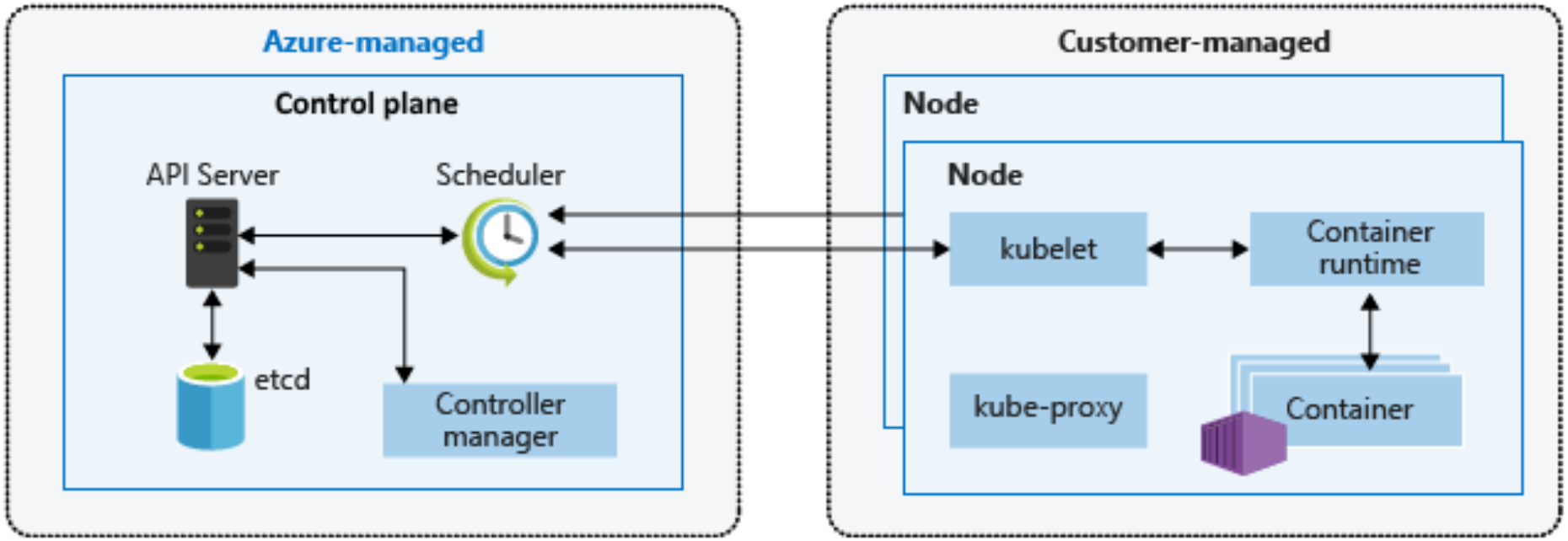
The cluster master is provided as a managed Azure resource abstracted from the user

The cluster master includes the core Kubernetes components like kube-apiserver, etcd, kube-scheduler and kube-controller-manager

An AKS cluster has one or more nodes, which is an Azure virtual machine (VM) that runs the Kubernetes node components and container runtime

https://docs.microsoft.com/en-us/azure/aks/

Azure Kubernetes Service



Demo

RUNNING A CONTAINER IN AZURE KUBERNETES SERVICE

Azure service fabric

Service Fabric is a microservices platform for building applications

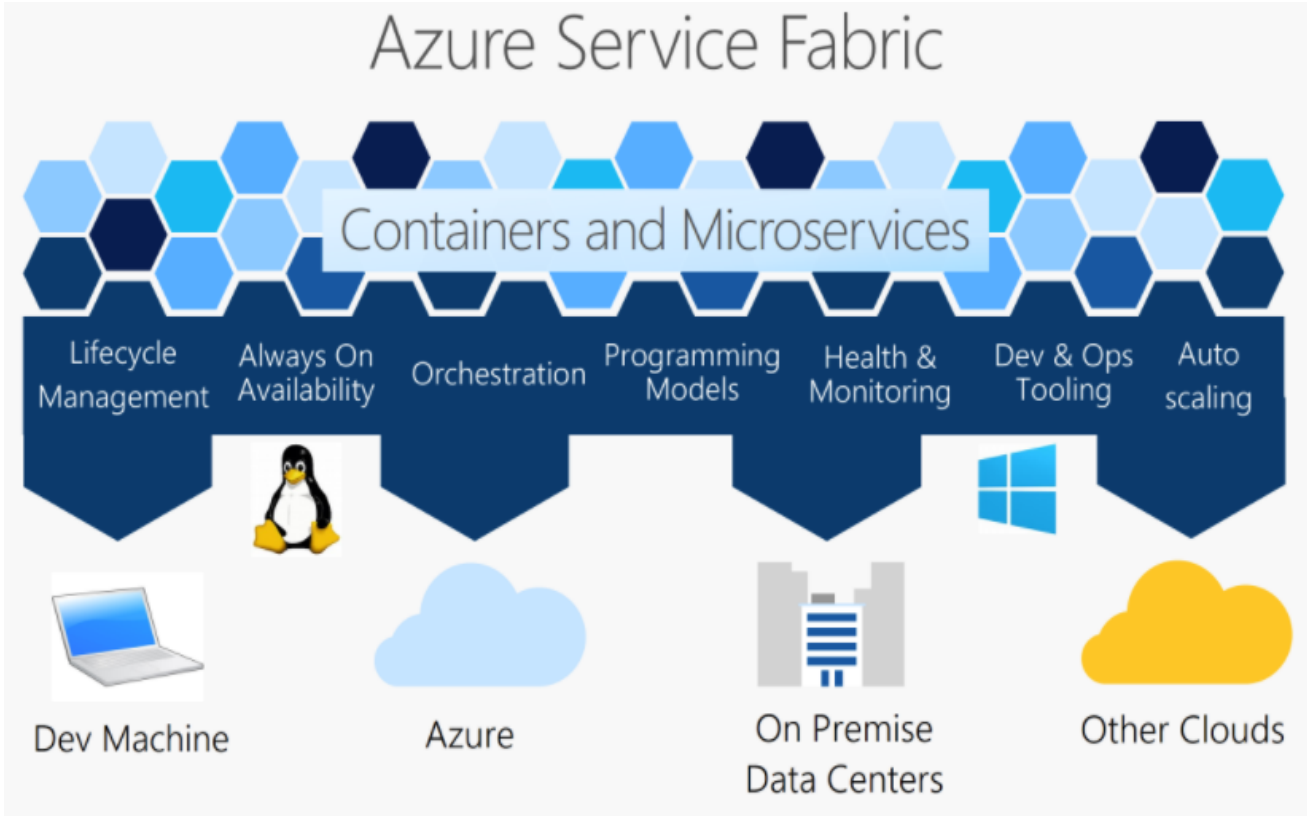
It is an orchestrator of services and creates clusters of machines

By default, Service Fabric deploys and activates services as processes, but Service Fabric can also deploy services in Docker container images

More important, services in processes can be mixed with services in containers in the same application

Supports different programming models, from using the Service Fabric programming models to deploying guest executables as well as containers

Service Fabric supports prescriptive application models like stateful services and Reliable Actors.



https://azure.microsoft.com/en-us/services/service-fabric/

Resources for further reading

1. Docker for Desktop - https://www.docker.com/products/docker-desktop
2. Docker Documentation - https://docs.docker.com/
3. Azure Container Registry - https://docs.microsoft.com/en-us/azure/container-registry/
4. Azure Container Instances - https://docs.microsoft.com/en-us/azure/container-instances/
5. Azure Web Apps for Containers - https://azure.microsoft.com/en-us/services/app-service/containers/
6. Azure Kubernetes Service - https://docs.microsoft.com/en-us/azure/aks/intro-kubernetes
7. Azure Service Fabric - https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-containers-overview
8. GitHub repo for dotnet-docker samples - https://github.com/dotnet/dotnet-docker


thank
you