Microsoft

Modernizing Applications with Containers and Orchestrators

Microsoft Services

Agenda

Day 1

Introduction to Containers

Getting Started with Windows Containers

Advanced Docker Topics

Day 2

Microservices and Containers

Container Orchestrators

Day 3

DevOps with Containers

Monitoring and Troubleshooting Containers

You Trainer

Radhika Jayaprakash

Microsoft Customer Engineer

Software Engineer/Developer for more than 15 years

Specialized in Java, Database, Containers and DevOps

Live in Ottawa with my husband and 2 kids

Your Role/Experience

Please introduce yourself:

- About you :
  - What is your role in your organization? (Developer, Operations, Manager etc.)
  - Technologies you work on
- Your experience with Docker / Containers , Kubernetes, Azure
- Expectations from this workshop

Microsoft

Module 1 - Introduction to Containers

Microsoft Services

Objectives

- Understand what Containers are
- Learn Docker Fundamentals (Docker Engine and Client)
- Understand Container Images and Docker Registry
- Learn How to Build Container Image using Dockerfile
- Learn how to Start, Stop, and Remove Docker Containers
- Understand use of Tags for Versioning Images

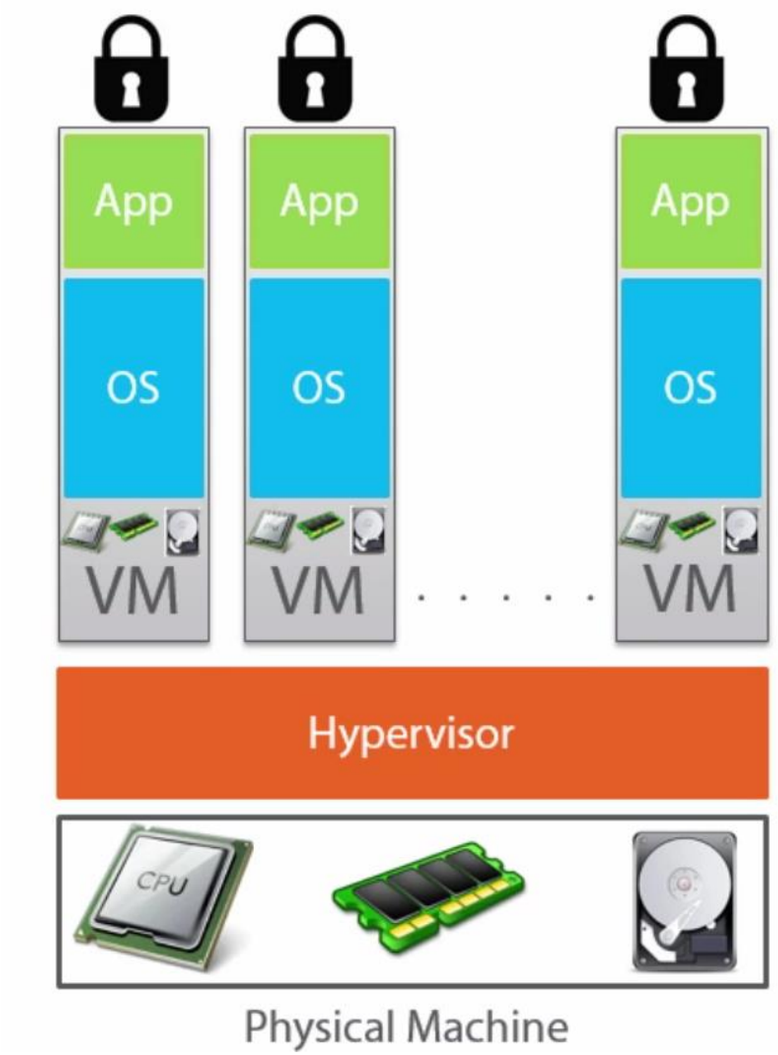
Why Docker?

- Build any app in any language using any stack (OS)
- Dockerized apps can run anywhere on anything
- No more “It works on my machine”
- No more dependency daemons so Developers and System admins unite

Docker Vocabulary

Host	A VM running the Docker Daemon to host a collection of Docker Containers
Client	Where docker commands are executed (client/server)
Image	An *ordered collection of filesystems (layers)* to be used when instancing a container (more on it later)
Container	A runtime instance of an image
Registry	A collection of docker images

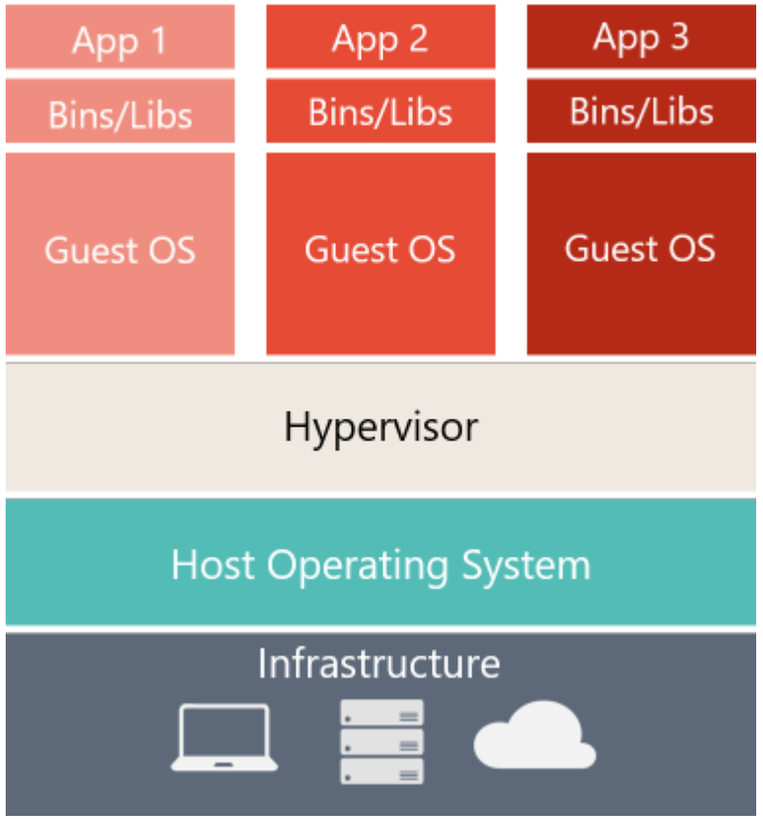
Challenges with Virtualization



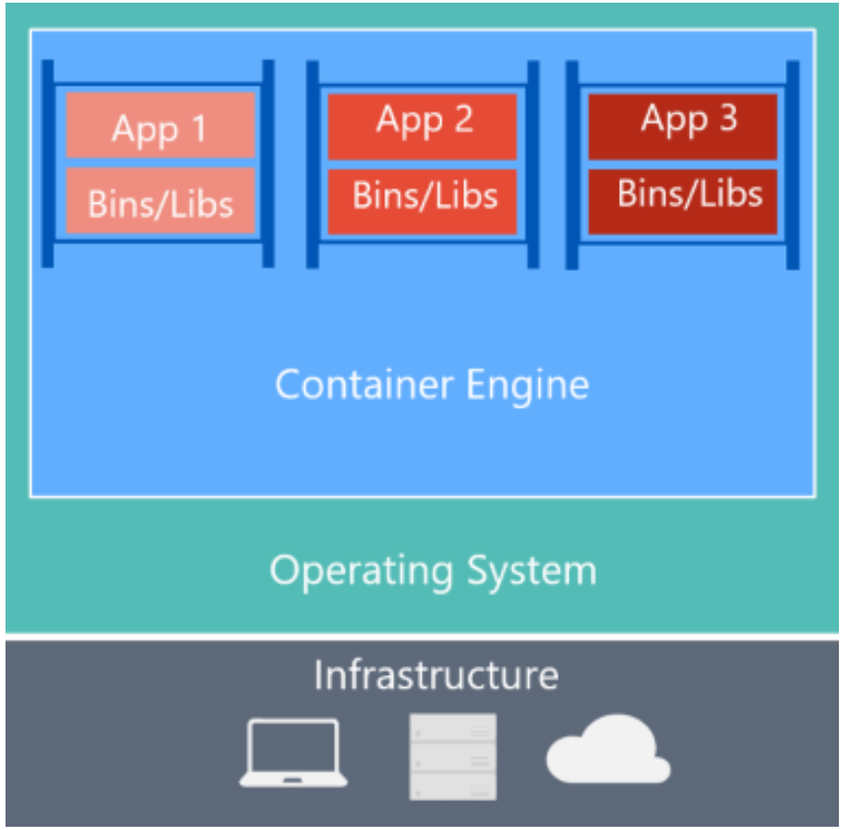
> OS != Business Value

Virtual Machines versus Containers

Virtual Machine

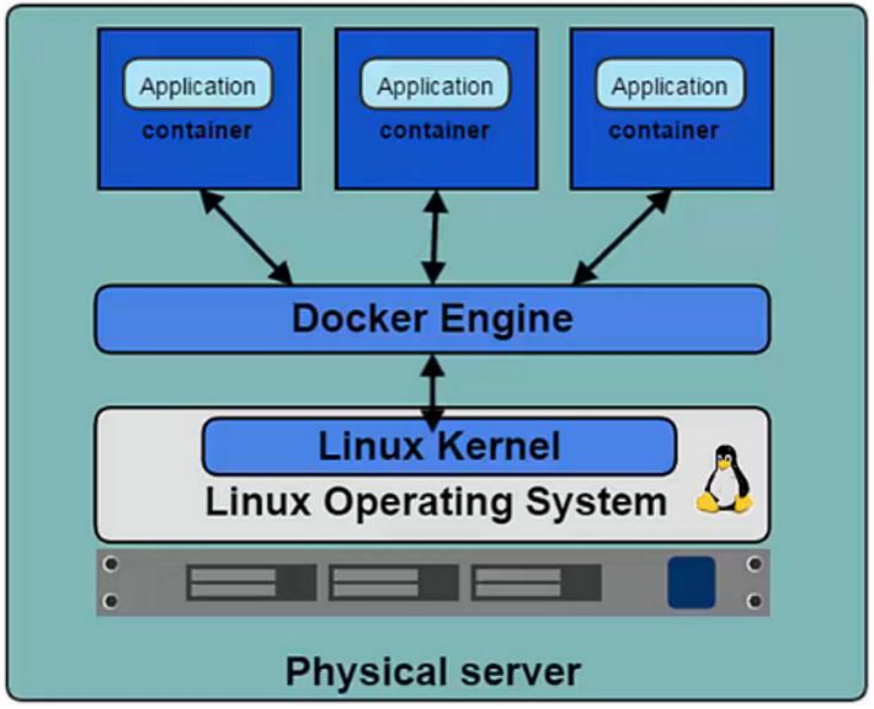


Container



Docker Platform

- Docker Engine (a.k.a. Docker Daemon)
  - The program that enables containers to be built, shipped, and run.
  - Uses Linux Kernel namespaces and control groups to give an isolated runtime environment for each application
- Docker Hub
  - A online registry of Docker images
- Docker Trusted Registry
  - Private on-site Registry for Docker images

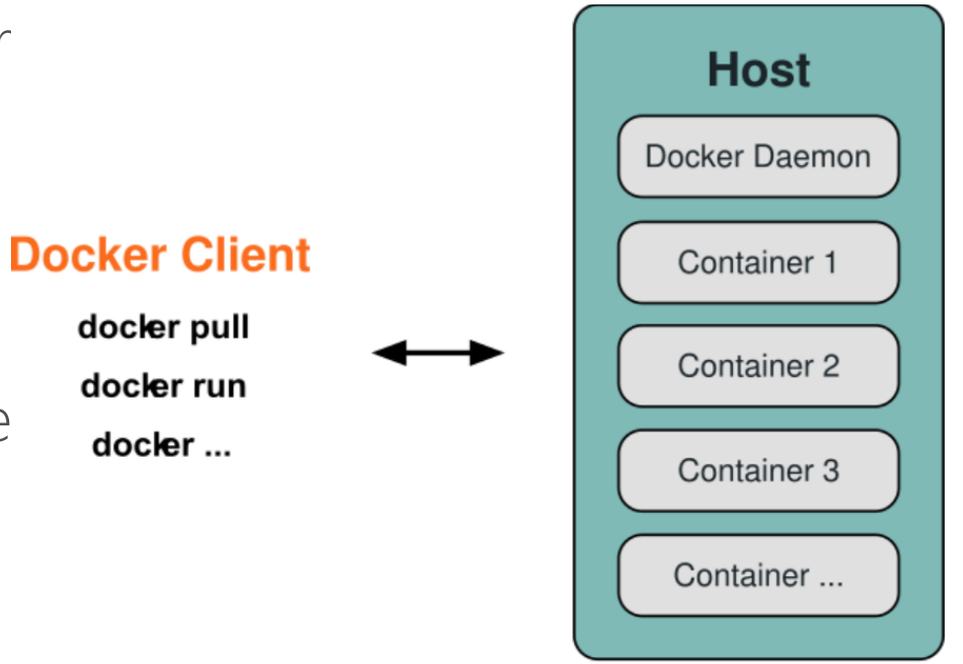


Docker Platform (Cont.)

- Docker Client
  - Takes user inputs and sends them to the Daemon.
  - Client and Daemon can run on the same host or on different hosts.

- Docker Images
  - Read-only template used to create containers.
  - Contains a set of instructions for creating the containers.

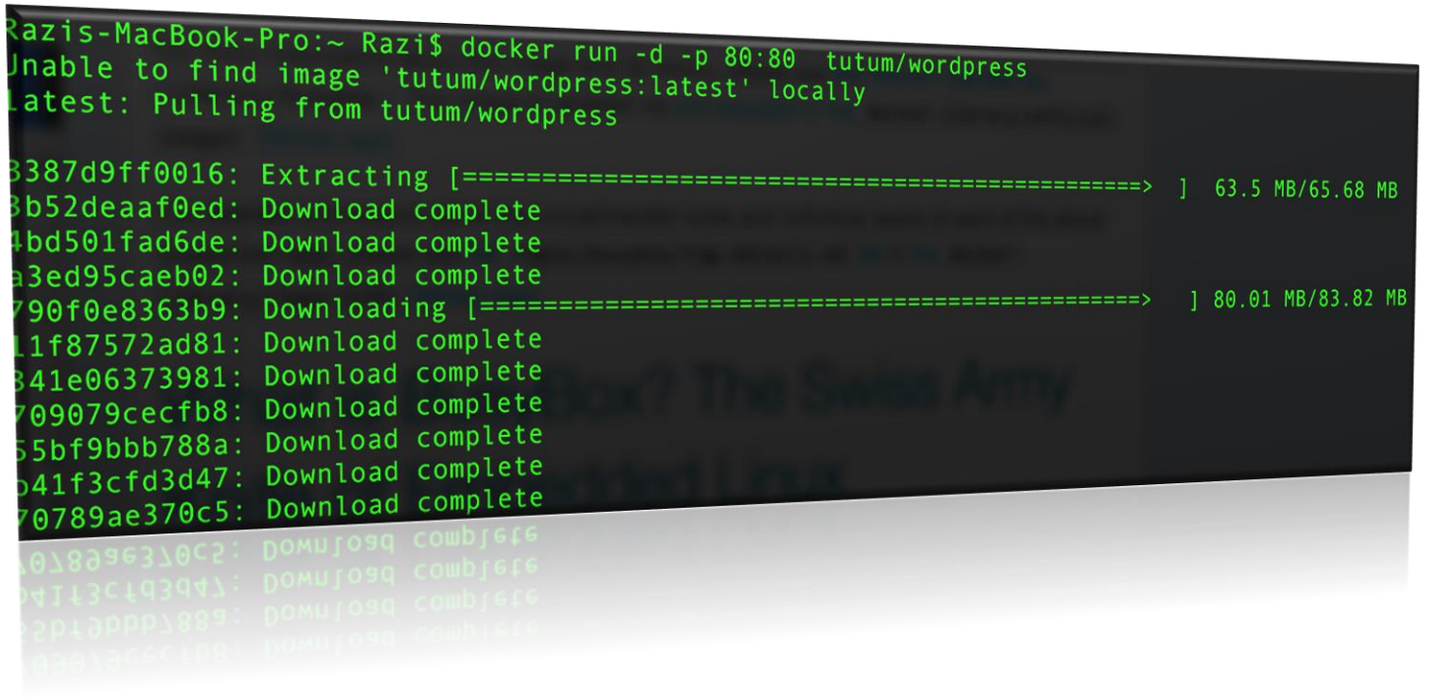
- Docker Containers
  - Isolated application platform based on one or more images.
  - Contains everything needed to run your application.



Quick Question?

How fast you can launch a fully functional WordPress blog engine?

How about multiple WordPress blog engines running side by side on same host?

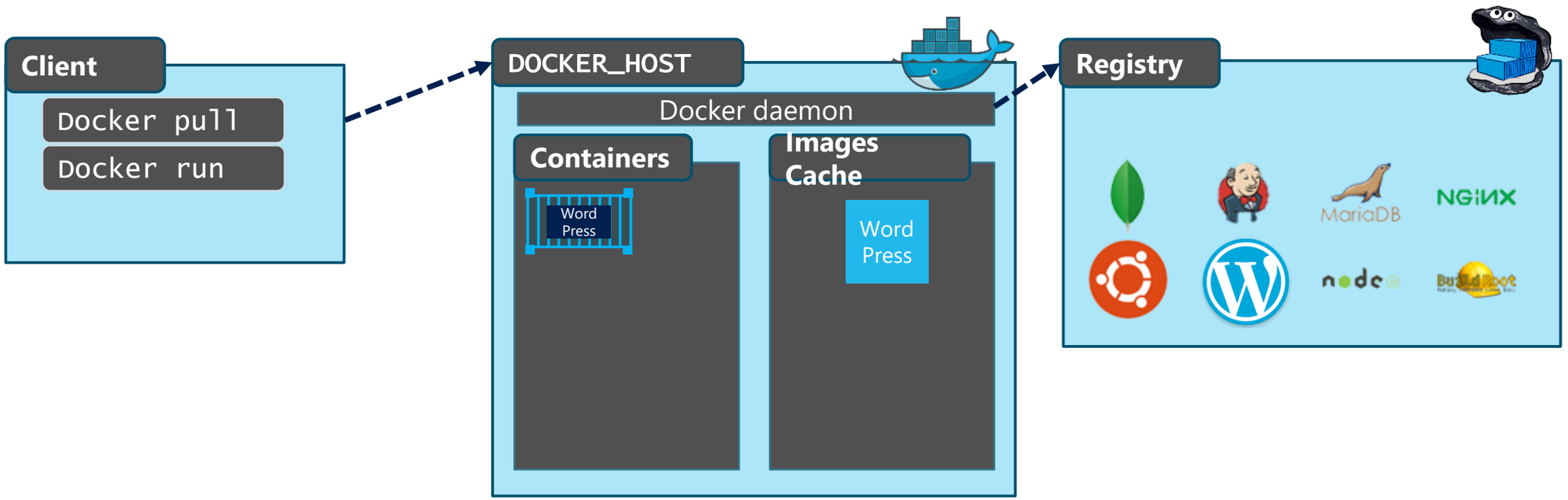


Demonstration: Running Docker Containers

Launch a single WordPress Container

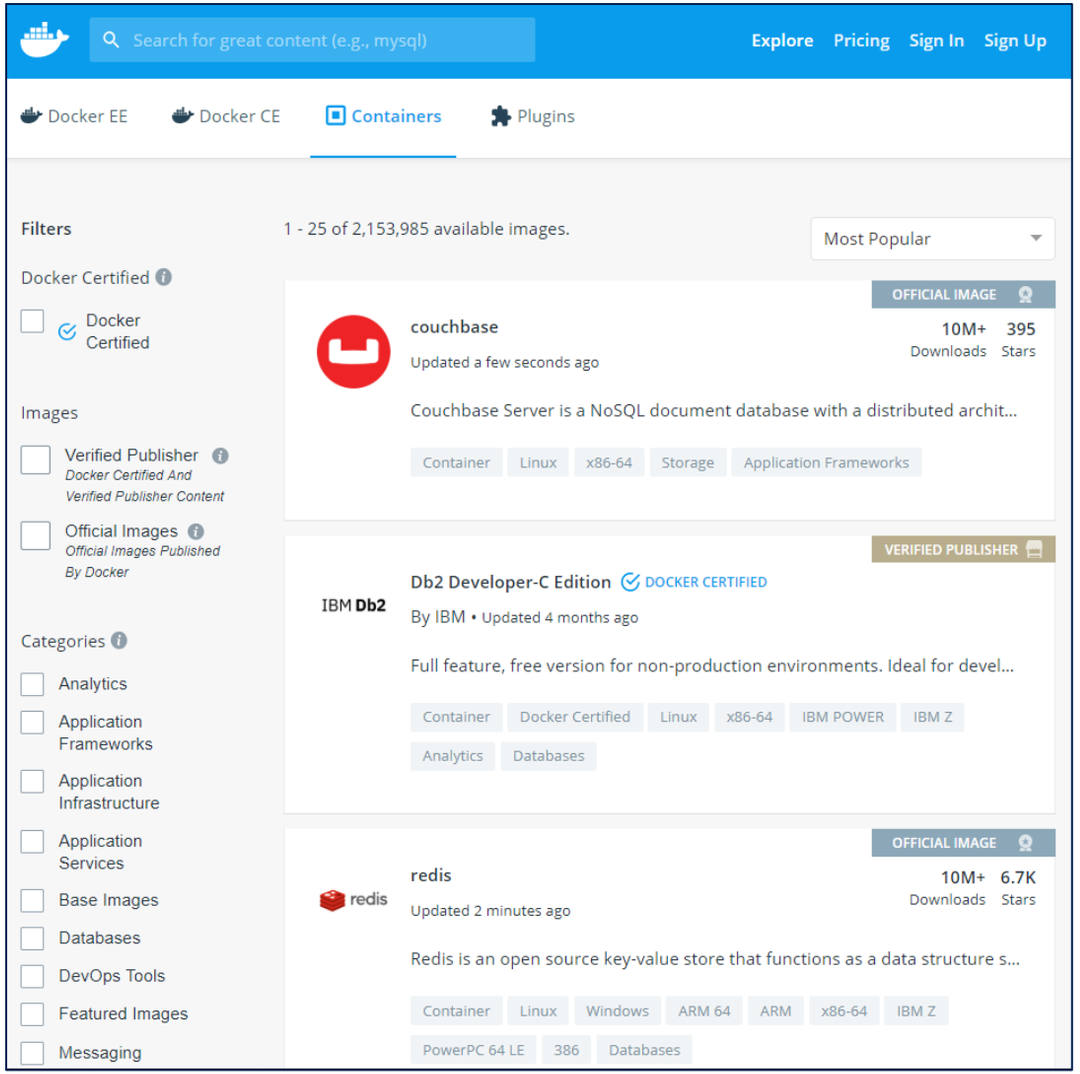
Running multiple WordPress Containers side by side

Docker In Action



Docker Registry

- Stores docker images
- Searchable
- Public Registry – hub.docker.com
- Private Registries – Instanced for you. E.g. Azure Container Registry
- The Registry is open-source under the permissive Apache License



Demonstration: Docker Registry

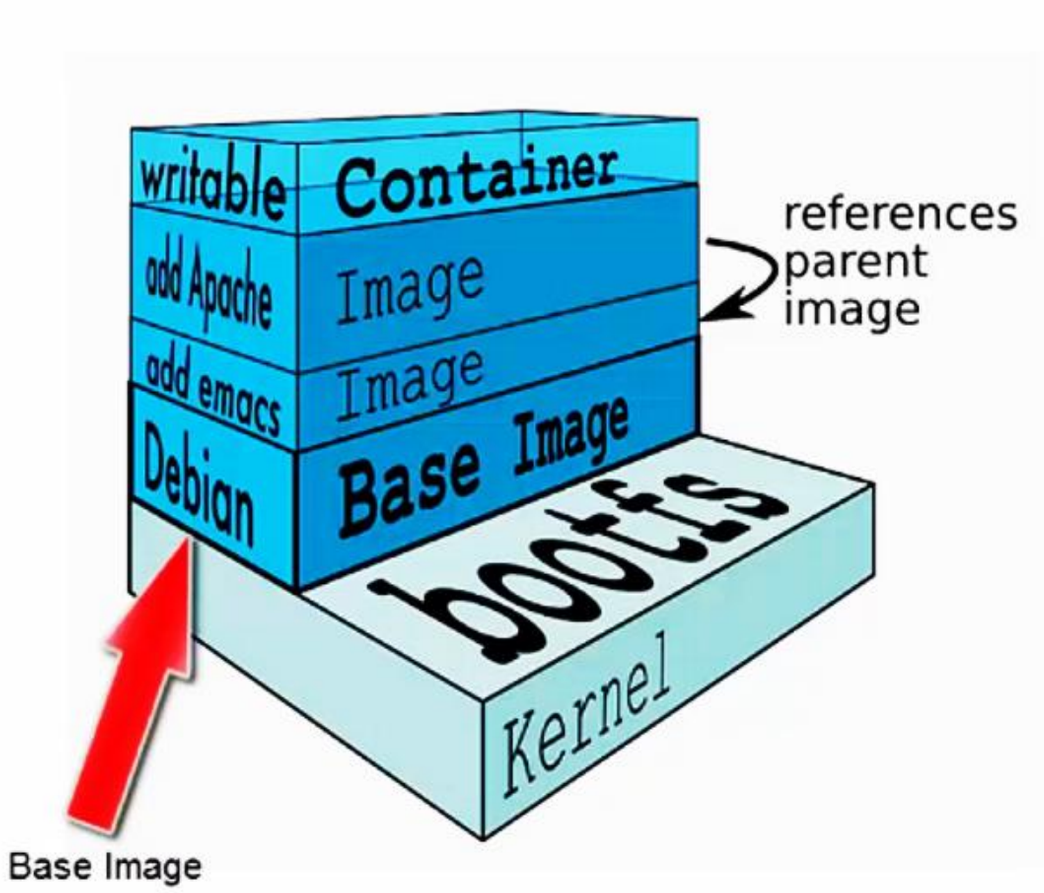
Search Docker Registry using Docker CLI

Search Images on DockerHub

Docker Image Naming Convention

Docker Images

- A Docker image is built up from a series of layers.
- Base platform OS image is provided by vendors like Microsoft for Windows OS image, Canonical for Ubuntu image etc. These images get published to DockerHub.
- Each layer represents an instruction in the image's Dockerfile.
- Each layer except the last one is read-only.



Demonstration: Docker Image Layers

List All Layers for Docker Image

Look at locally cached images

Dockerfile

- Text file with Docker commands in it to create a new image. You can think of it as a configuration file with set of instructions needed to assemble a new image.

- Docker has a docker build command that parses Dockerfile to build a new container image.

```
# Simple Dockerfile for NGINX

FROM nginx:stable-alpine

MAINTAINER Razi Rais

COPY index.html /usr/share/nginx/html/index.html

CMD ["nginx", "-g", "daemon off;"]
```

```
FROM microsoft/dotnet:1.1.0-sdk-projectjson

COPY . /app

WORKDIR /app

RUN ["dotnet", "restore"]

RUN ["dotnet", "build"]

EXPOSE 5000/tcp

CMD ["dotnet", "run", "--server.urls", "http://*:5000"]
```

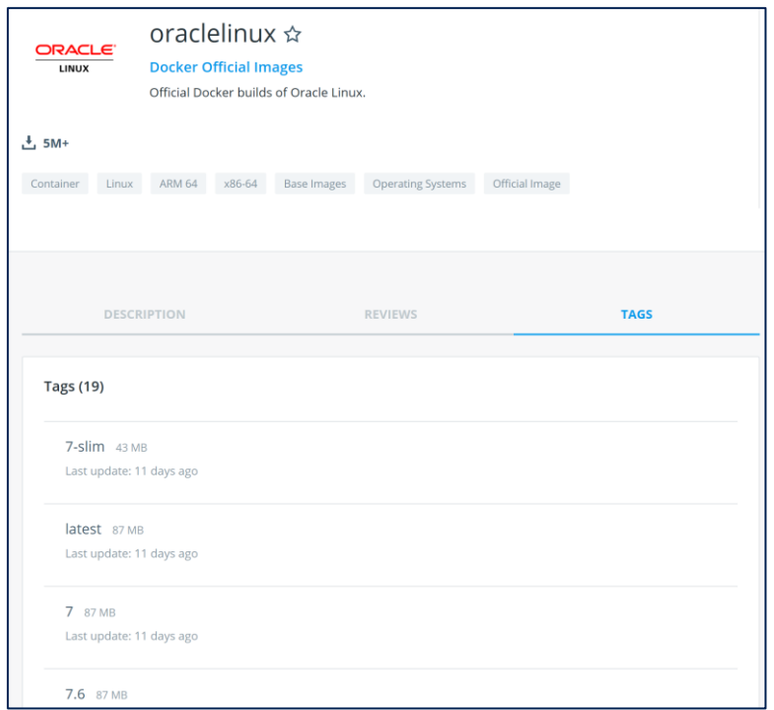
```
# Simple Dockerfile for NodeJS

FROM node:boron

MAINTAINER Razi Rais

# Create app directory
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

# Install app dependencies
COPY package.json /usr/src/app/
RUN npm install

# Bundle app source
COPY . /usr/src/app

EXPOSE 8080

CMD [ "npm", "start" ]
```

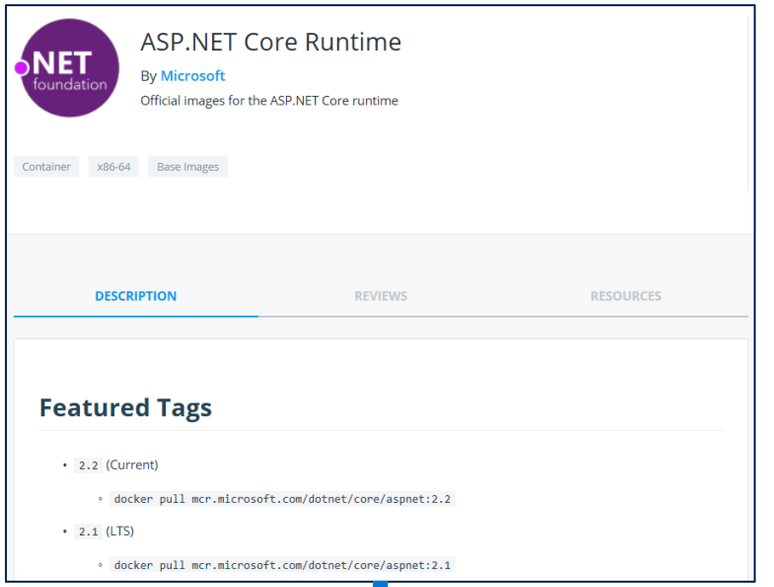
Common Dockerfile Instructions

- FROM instruction initializes a new build stage and sets the Base Image for subsequent instructions.
- LABEL is a key-value pair, stored as a string. You can specify multiple labels for an object, but each key-value pair must be unique within an object.
- RUN will execute any commands in a new layer on top of the current image and commit the results.
- WORKDIR instruction sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it.
- ADD instruction copies new files, directories or remote file URLs from <src> and adds them to the filesystem of the image at the path <dest>.
- COPY instruction copies new files or directories from <src> and adds them to the filesystem of the container at the path <dest>.
- CMD provide defaults for an executing container. These defaults can include an executable.
- ENTRYPOINT allows you to configure a container that will run as an executable.
- EXPOSE instruction informs Docker that the container listens on the specified network port(s).

Image Tags

- A Tag name is a string value that you can use to distinguish versions of your Docker images so you can preserve older copies or variants of a primary build.
- You can group your images together using names and tags (if you don't provide any tag default value of latest is assumed)





Demonstration: Dockerfile and Docker Build

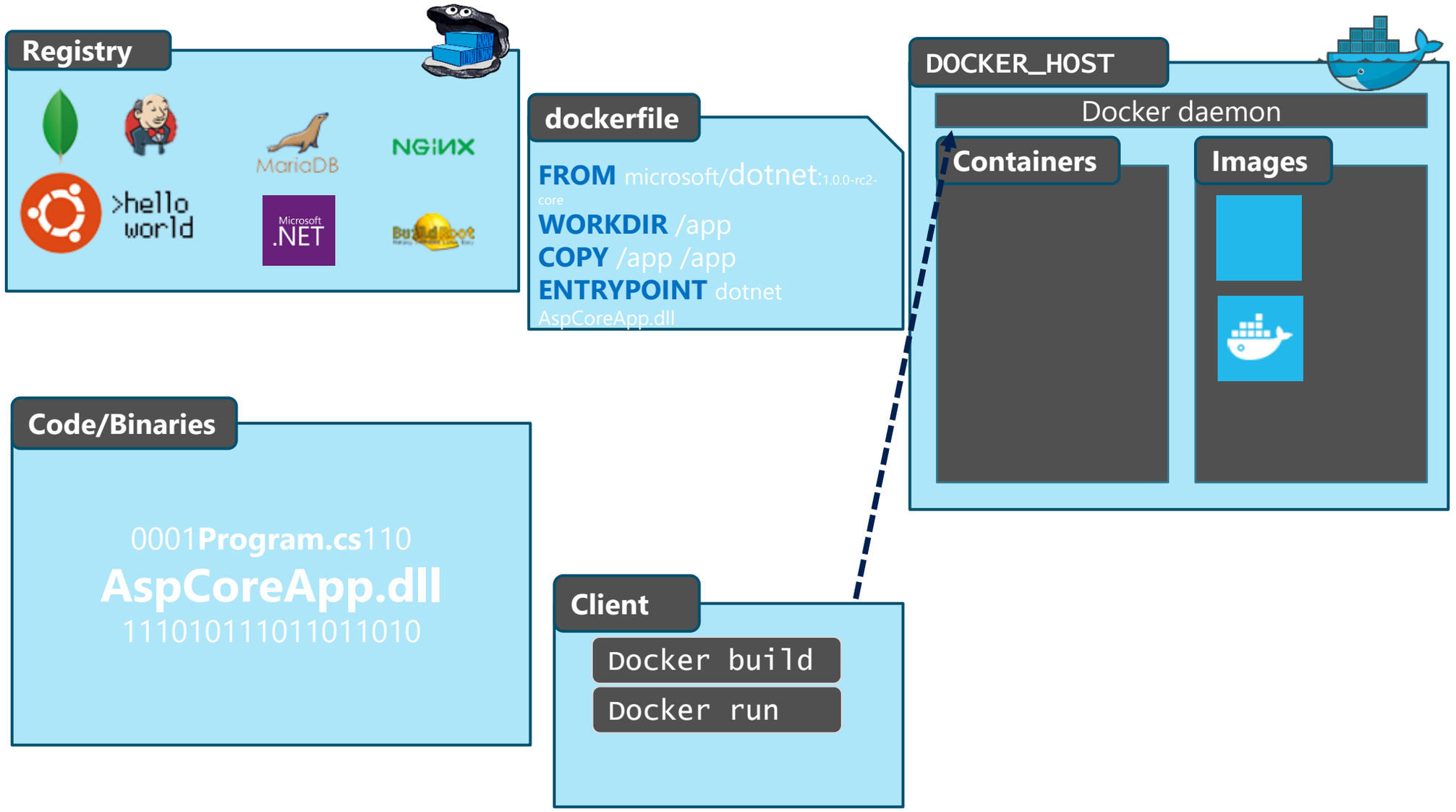
Working with Dockerfile

Build container images using Docker build command:

- NodeJs
- Nginx
- ASP.NET Core

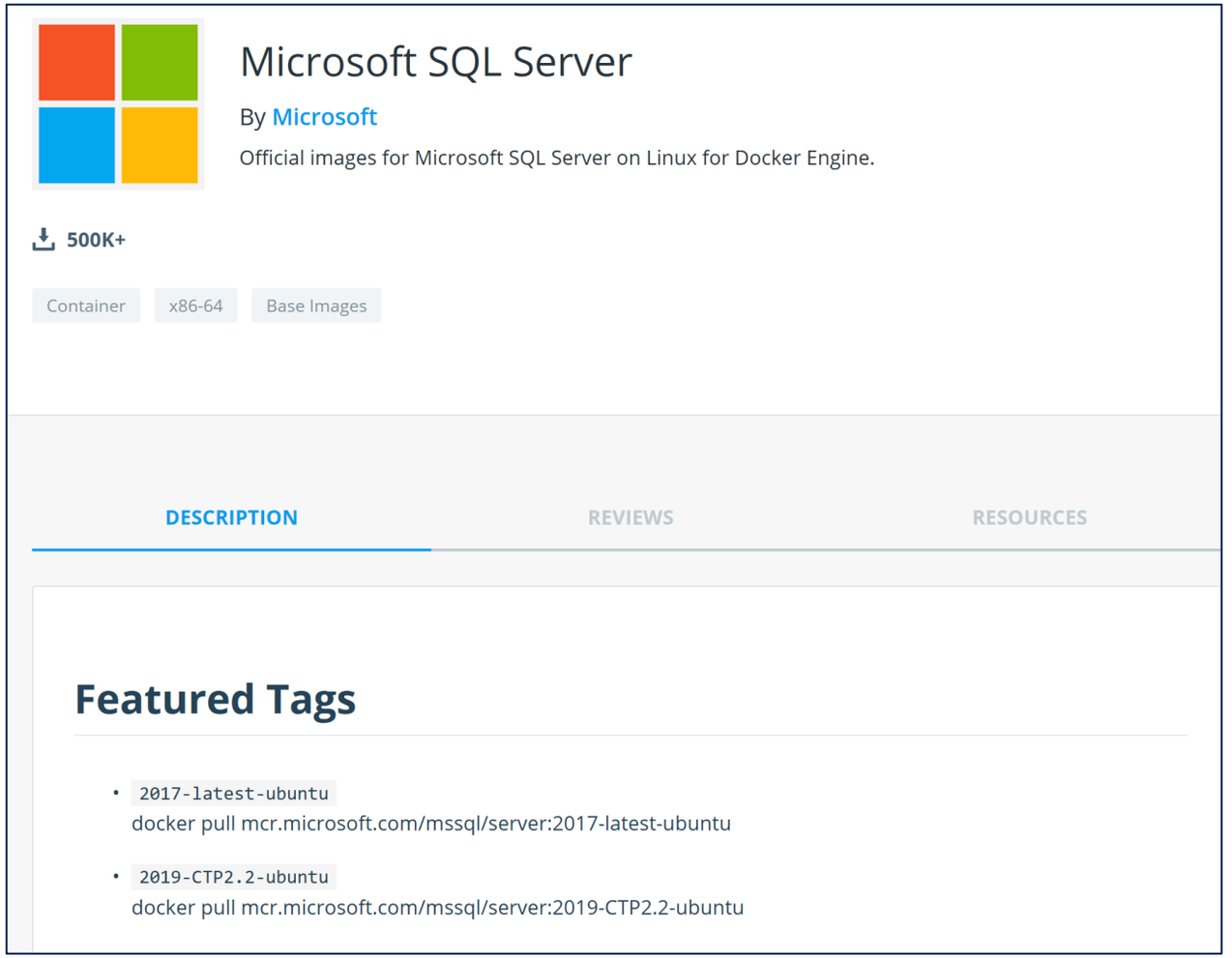
Using Image Tags

How does Docker build work?



SQL Server 2017 Container Image

- SQL Server in Linux can be packaged in a Docker container
- Can be used in automated tests to pre-populate a SQL Server instance with test data on-demand
- Can run on top of Ubuntu 16.04, RHEL and CentOS



Demonstration: *Running SQL Server 2017 inside Container*

SQL Server 2017 Container Image

Using Custom Database with SQL Server 2017 running inside a Container

Knowledge Check

- What is docker registry ?

  A Docker registry is a storage and distribution system for named Docker images

- Why should we tag images ?

  A tag conveys information such as the base OS version, or whether it is the most recent update of the image. In this way, tags help developers and admins differentiate between the various available images in a repository.

- How do we run container in detached (background) mode ?

  docker run -d --name myredis -p 6379:6379 redis

Lab: Introduction to Containers

Running Your First Container

Working with Docker Command Line Interface (CLI)

Building Custom Container Images with Dockerfile

Interaction with a Running Container

Tagging

Setup Labs

http://aka.ms/PremierEducation

Sign in with a Microsoft Account (Live/outlook or personal ID).

Navigate to **: WorkshopPLUS -> My Training -> Redeem Training Key**.

Training Key: **14AF9E739B574ABF**

Launch Linux (Ubuntu) VM Labs (UPD20211105)