Developers, Containers and the Cloud

Course Map

1. **Developers, Containers, the Cloud**
2. Getting Started with Docker
3. Exercise: Installing and using Docker
4. Working with Dockerfiles and ASP.NET Core apps
5. Exercise: Dockerfiles, ASP.NET Core apps

Course Map

6. Exercise: Mapping to Source Code
7. Exercise: Mapping to Shared Data
8. Multi-Container Applications with Docker Compose
9. Exercise: Linking vs networked Containers, using Docker Compose

Agenda

- Cloud Computing and Microservices
- Introduction to DevOps
- Docker, Containers, Virtual Machines
- Windows Containers, Hyper-V, Nano Server
- Management and orchestration Tools

Get the Bits

github.com / tonysneed /

Kliant.DockerForDevs

What is the Cloud?

Why should you care?

- Computing resources allocated on a **pay-as-you-go** basis

- **Efficiency** is important: disk I/O, memory, CPU

Microservices

- Microservices represent **self-contained** units of functionality

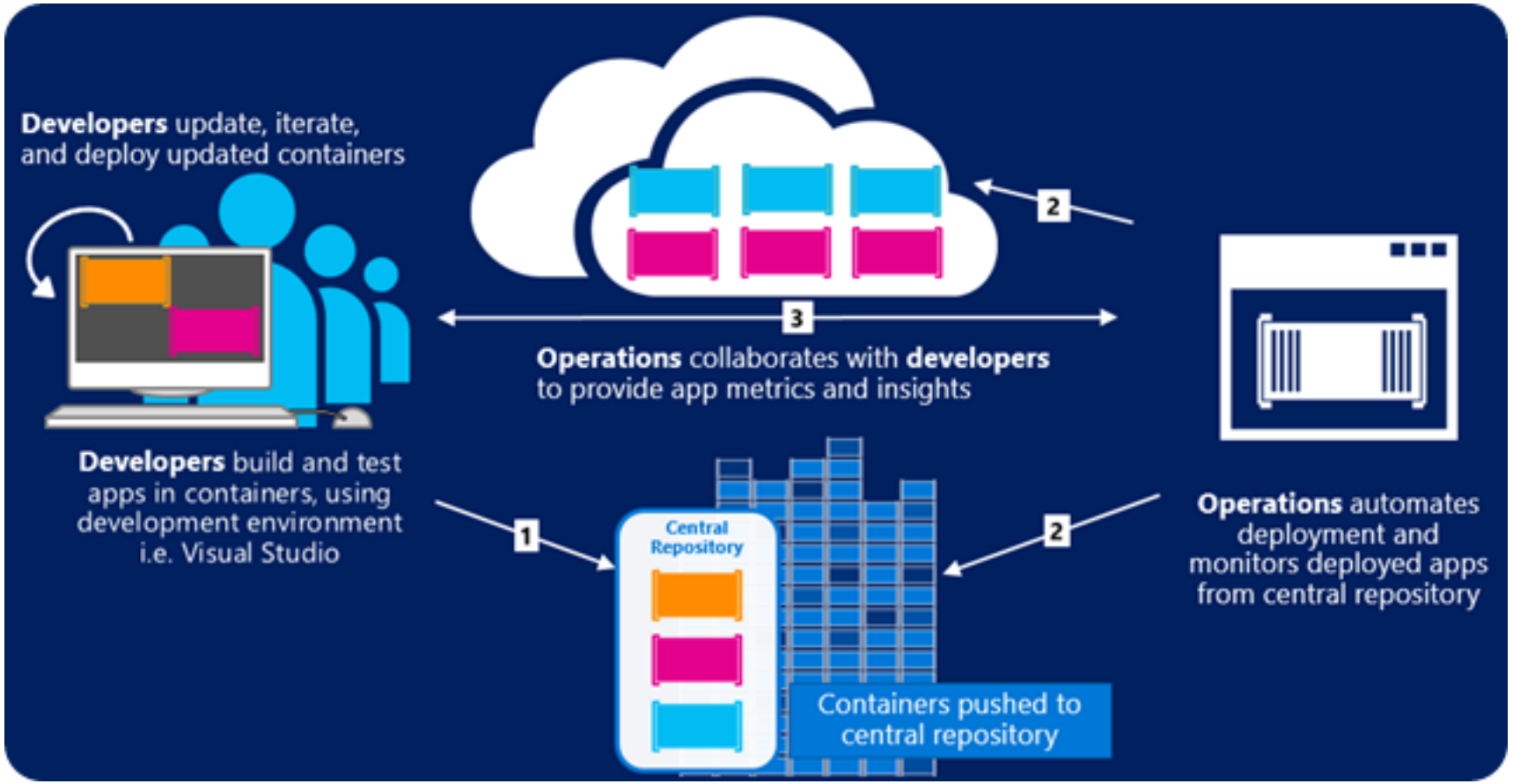
Microservices

- Loosely coupled dependencies on other services
- Independently updated, tested and deployed
- Scaled independently of other services
- Fault tolerant, highly available

Intro to DevOps

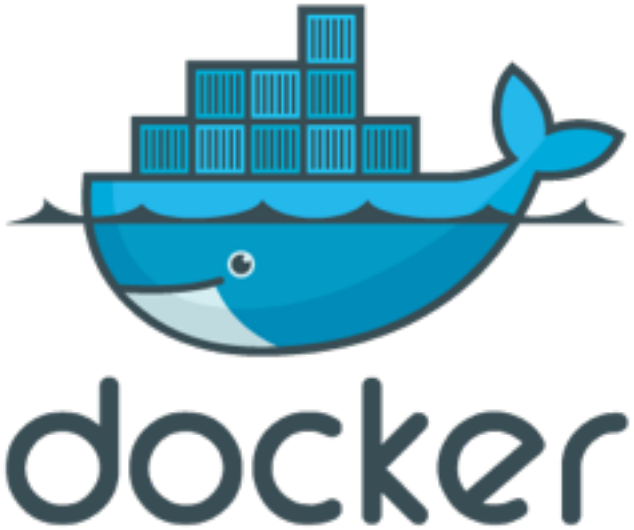
- **Developers** and **IT pros** working *together*
- **Toolchain:**
  - Code
  - Build
  - Test
  - Package, Release
  - Configure, Monitor

DevOps and containers



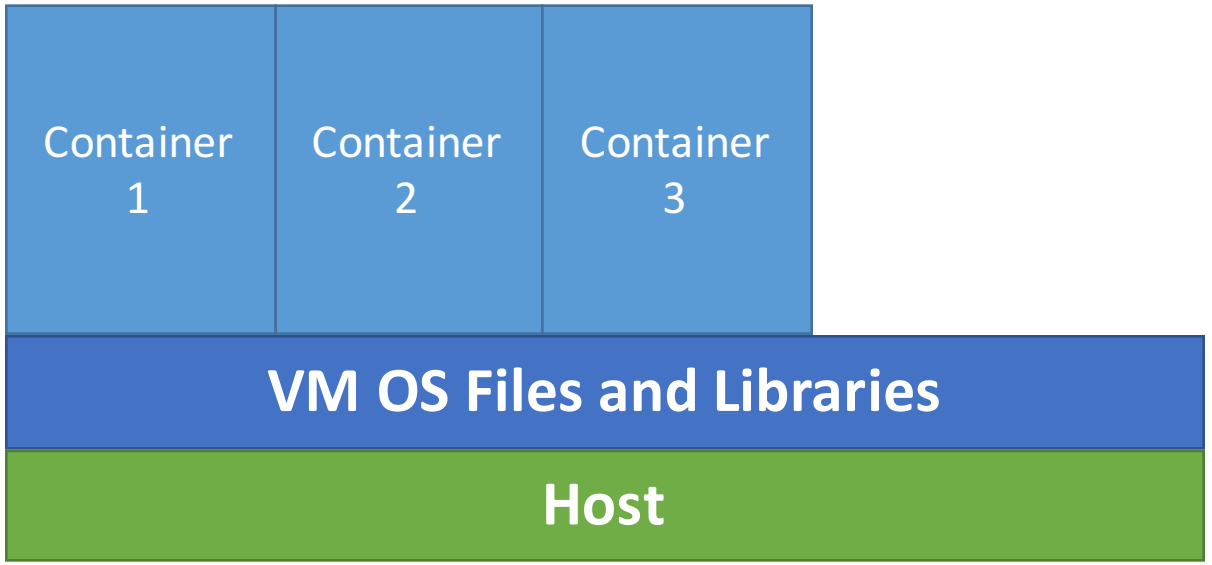
Docker containers

- Docker containers wrap up a piece of software in a **complete filesystem** that contains *everything it needs to run*



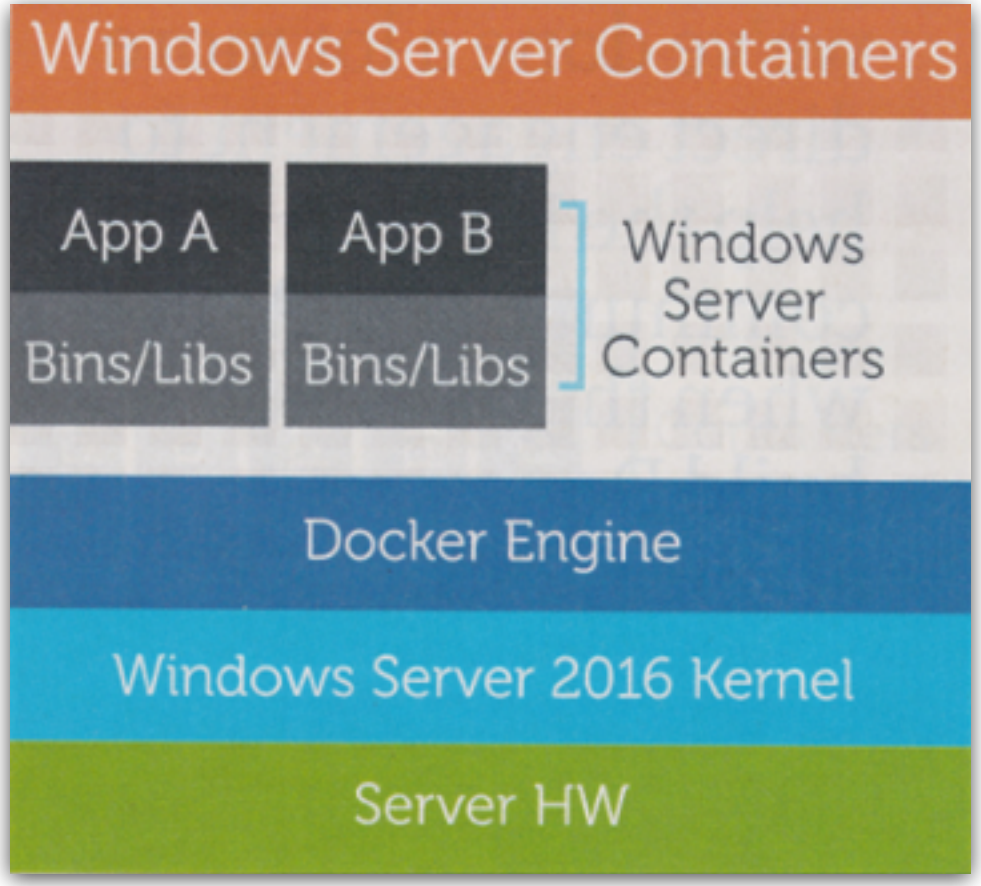
Containers vs virtual machines

- Containers provide **isolation** like VM's, but without the **overhead** because of shared OS resources



Windows Server Containers

- Windows containers also provide **isolation** while sharing the OS kernel
- **Hyper-V** containers provide *increased* isolation but carry additional *overhead*
- **Docker** can be used to manage Windows containers



Nano Server

- **Lightweight** version of Windows Server
- Compatible with **.NET Core**
  - But not the *full* .NET Framework

Docker Orchestration Tools

- Docker **Compose**
  - Link multiple containers
- Docker **Cloud**
  - Deploy stacks of services
- Docker **Swarm**
  - Scaling, high availability

Questions?