Docker Concepts

Tuesday, April 21, 2020 2:36 PM

Scaling out

- run more instances of your app to handle increased traffic
    - more users trying to connect… another instance so your website is faster

    - but decreases isolation without virtual machines
        - bc instances of the program can compete to serve the users

- how to scale out:
    1. set up a new physical server
        - slow to set up, inflexible
            - couple of hours, new windows machine, set up IIS, configure, sign in to accounts, download code package assemblies

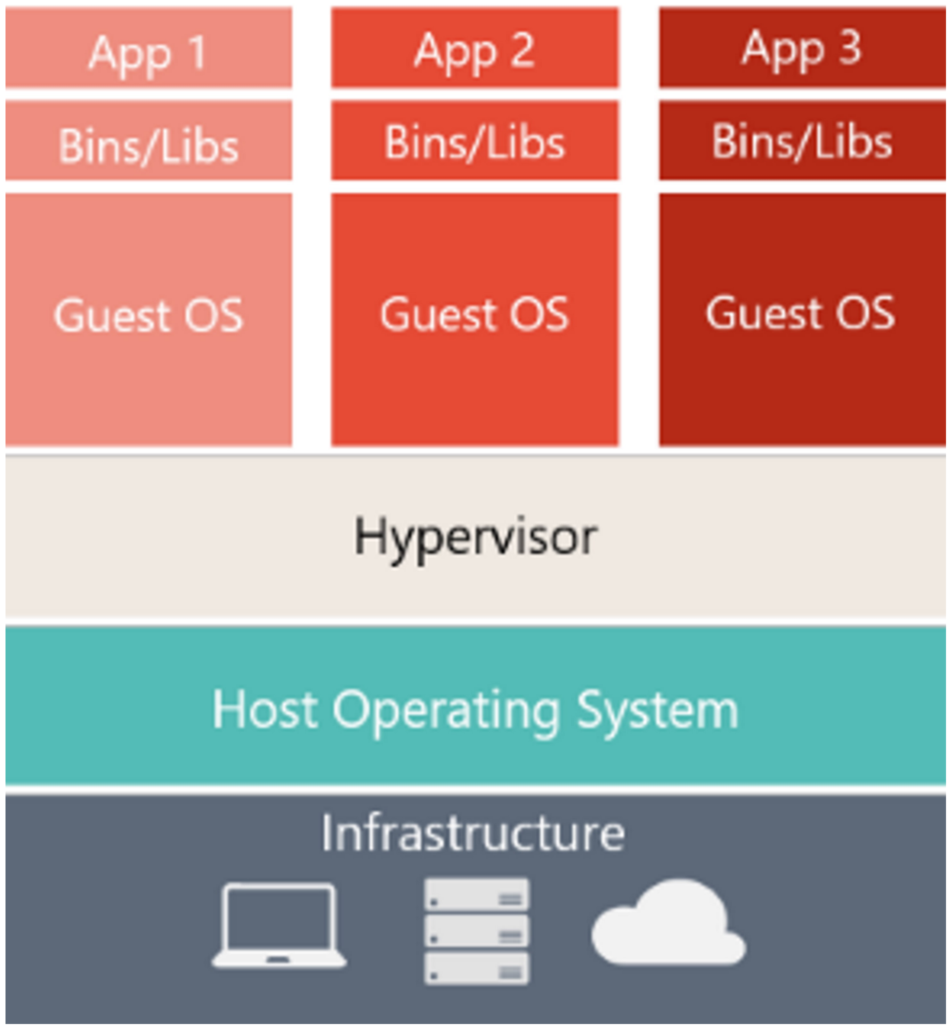
    2. virtual machine
        - on premises or on cloud

        - more flexible and faster to set up (a few seconds if you have a good image)
            - flexibility is good for spikes in usage during ex the christmas season

            - so servers usually on virtual machines

        - faster boot up

        - Disadvanatages
            - wasted space (operating system takes up a lot of space



Infrastructure
- hardware
  - cpu, ram

Host Operating System

Hypervisor
- expose virtual hardware based upon real hardware accessed through the host machine OS

- prevent to be a non special hard disk and pretend to be a baseline that you can install an OS on
  - pretend ethernet connections/wifi

- VMWARE, VirtualBox, Hyper-V

Virtual machine installed on top of hypervisor
- for virtual machine

only the pretend/guest OS is accessible to you
and files and programs on the OS

standard OS (Windows) runs on the hypervisor for the virtual machine… so hypervisor has a hard job. It would be better if the virtual machine ran a special virtual machine version of the OS

very isolated environment
- lots of advantages

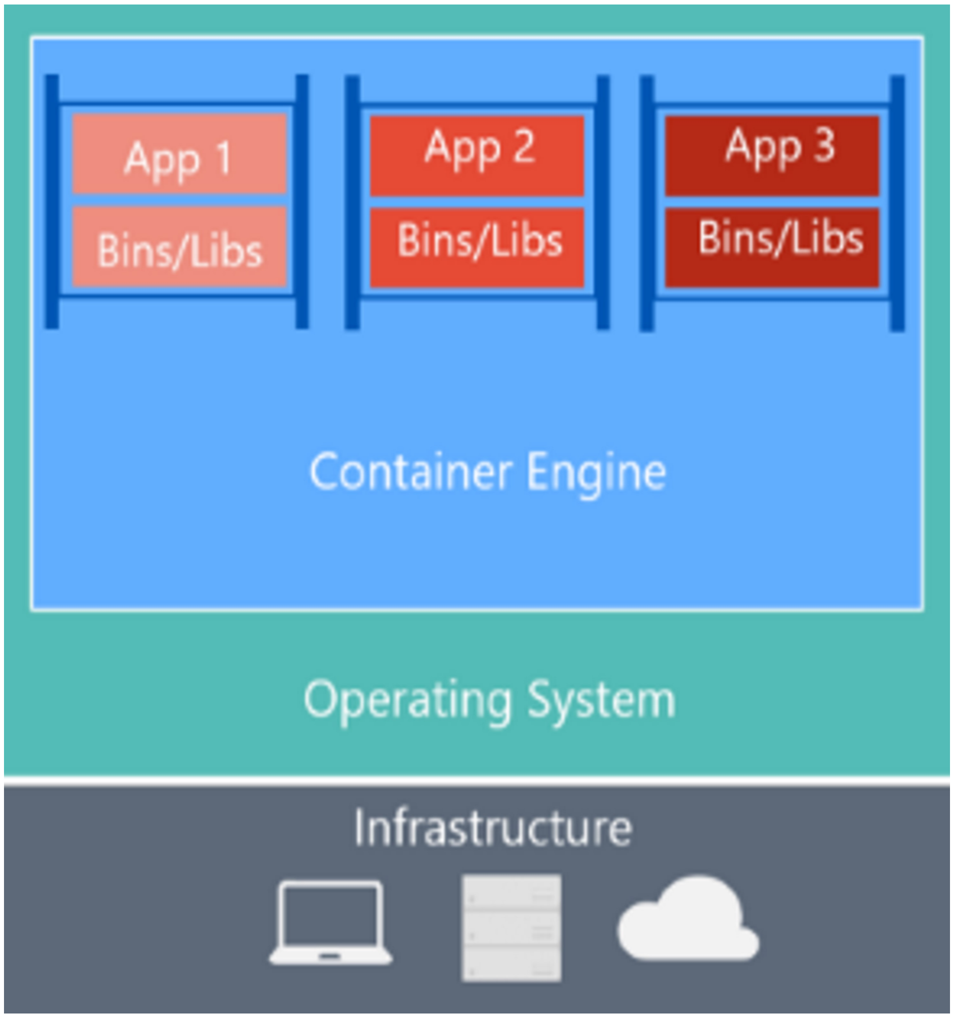
3. Containers
   - Only one operating system
     - for multiple containers
   - Most common container engine is Docker
   - Will provide isolation
     - not as much as virtual machine, but almost as good as virtual machines
     - Containers will look like it is the only application running on the machine
   - No overhead of the guest operating system
   - Flexibility
     - start and stop in milliseconds
       - virtual machine creation and boot time is longer
       - by comparison a container is instantaneous
     - more flexibility with the resources it uses
       - it doesn't grab on to a whole block of memory
       - memory that is not needed by the application is disallocated
       - like a process that runs on your computer
     - Makes it seem like it is its own operating system
       - doesn't conflict with other containers

Docker will manage the different containers
different Os's on one big machine

Container:

Running application with isolation provided by docker
technically some kind of virtual machine

Less redundancy
some shared bins and libs without having to install those dependencies twice



How do we make Docker Containers
containers are instances of Docker images

containers are ephemeral
quickly start up, sytop, get deleted , change their filesystem as they run. quick to disappear

- images are templates for starting a new container
    - every container comes from an image
    - images are immutable, persistent
        - Docker repositories and tags point to images
    - a VM image might be s of GB including the whole guest OS
        - can clone image to make copies of that virtual machine
        - make virtual machines quickly that are ready to go
    - a Docker image is much smaller
        - and using its layered filesystem approach can share files between images
- How do I make an image
    - Docker builds images using Dockerfile
    - Dockerfile is like a recipe for building the image, based upon the (1) base image it references and (2) a build context
        - a directory full of probably source code
    - image can make many different/distinct containers
        - same initial state, but can take different paths based on the input they execute and their code (and programs/filesystem)
    - Dockerfile can make different images based on the arguments given to it and based on the source code (given to the dockerfile)