Docker Concepts

Summary of Topics

- Why Docker containers
  - Simplifies hardware and operating system management
- Virtualization
  - Clear separation of hardware from software
    - E.g., Allows running Windows on a Mac computer
- Container defined
  - Run an app and its operating system in a process on any computer
- Docker terminology
  - Image, container, dockerfile

Why Containers?

Why containers?

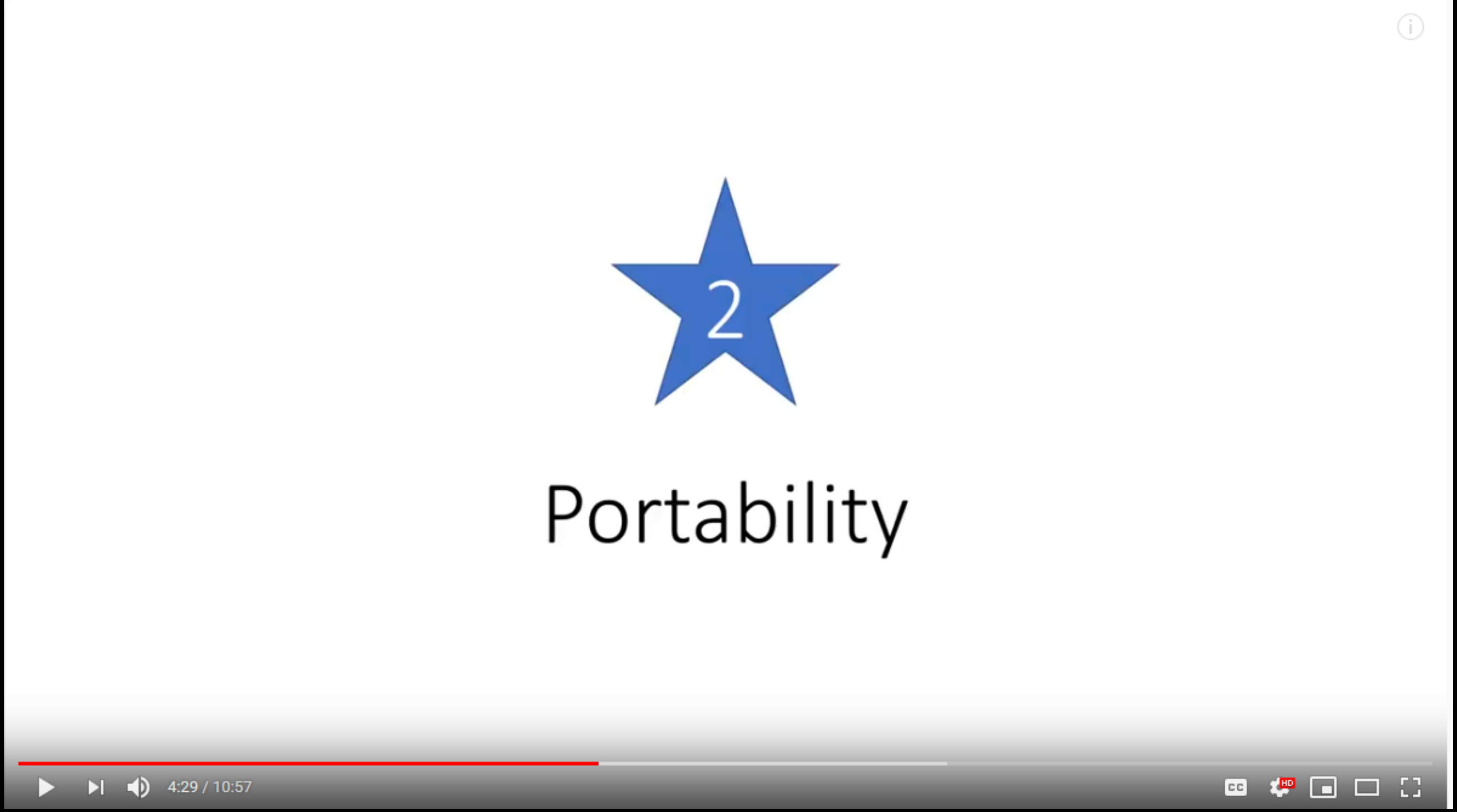
- Software will work wherever Docker is installed
- More software runs on less hardware
- Increased security
- Automates the build and deploy of software and hardware

- Infrastructure as code (IaC)
    - the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools
    - cost (reduction)
    - speed (faster execution)
    - risk (remove errors and security violations)

https://en.wikipedia.org/wiki/Infrastructure_as_code

Software will work wherever Docker is installed

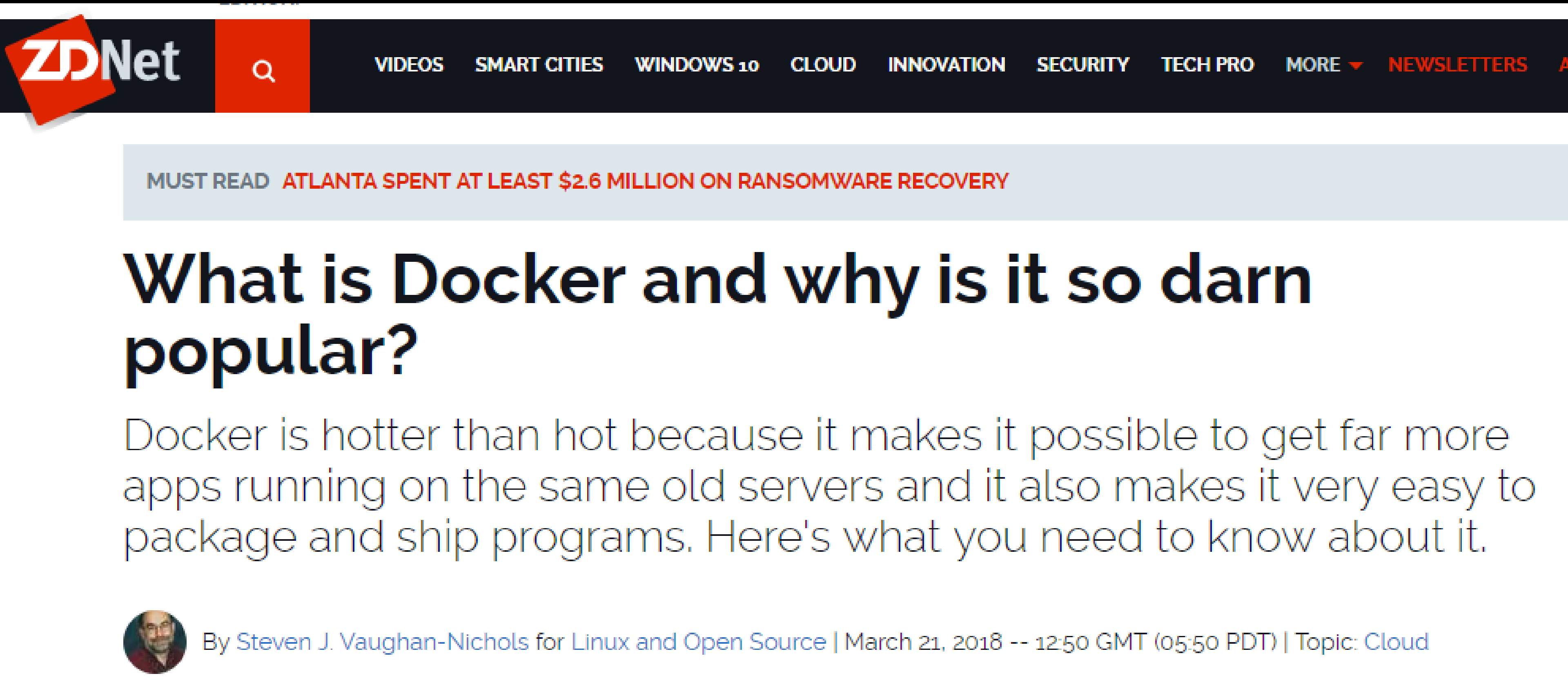
- Consider this development scenario



2

Portability

https://www.youtube.com/watch?v=YcqQI7jACTQ (4:25 - 7:50)



MUST READ ATLANTA SPENT AT LEAST $2.6 MILLION ON RANSOMWARE RECOVERY

What is Docker and why is it so darn popular?

Docker is hotter than hot because it makes it possible to get far more apps running on the same old servers and it also makes it very easy to package and ship programs. Here's what you need to know about it.

By Steven J. Vaughan-Nichols for Linux and Open Source | March 21, 2018 -- 12:50 GMT (05:50 PDT) | Topic: Cloud

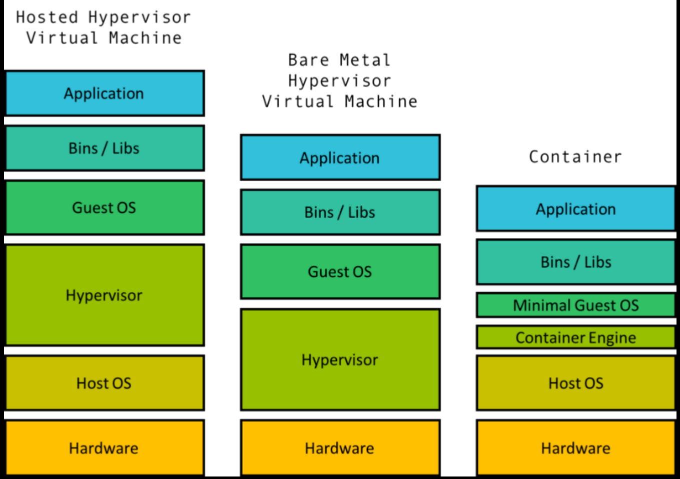
Cloud service layers

- **Infrastructure as a Service (IaaS)**: This model puts together infrastructures demanded by users, namely servers, storage, networks and datacenter fabric. The best IaaS examples are the AWS, GoGrid, Rackspace, Eucalyptus, flexscale, RightScale, etc.
- **Platform as a Service (PaaS)**: This model enables the user to deploy user-built applications onto a virtualized cloud platform. The best example of PaaS platforms are Google AppEngine, Windows Azure, Force.com, etc.
- **Software as a Service (SaaS)**: This refers to browser-initiated application software delivered to thousands of paid cloud customers. The best SaaS examples are Cloudera, Hadoop, salesforce.com, .NETService, Google Docs, Microsoft Dynamic CRM Service, SharePoint service, etc.

More software runs on less hardware

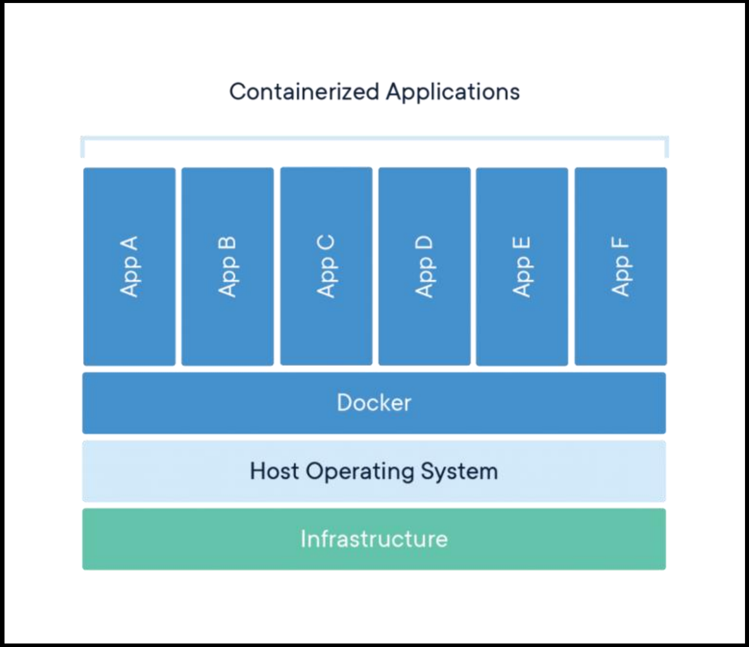
- Uses fewer resources than other virtual environments

- Used in managed clusters, where idle services can have minimal resources until they are scaled up



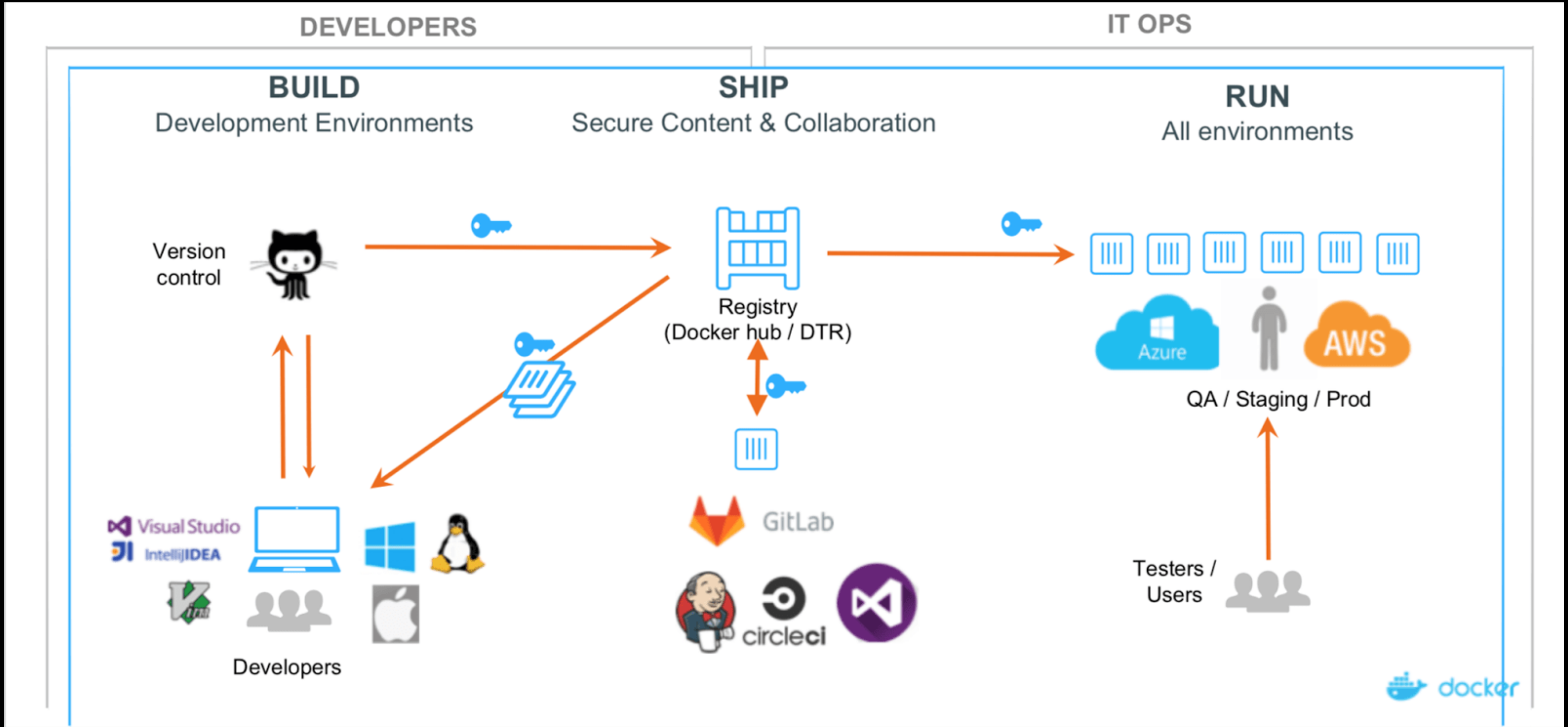
Container isolation

- Each container is isolated from each other using Linux primitives
  - Kernel namespaces
    - Each container in its own network stack
  - Control groups
    - Each container has a limited share of memory, CPU, disk I/O
- Containers run services, which communicate
  - Easier to develop with services
  - Can mix and match kinds of apps, frameworks, environments
    - Windows and Linux containers

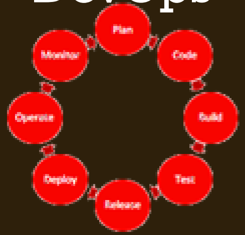


Automates the build and deploy of software and hardware

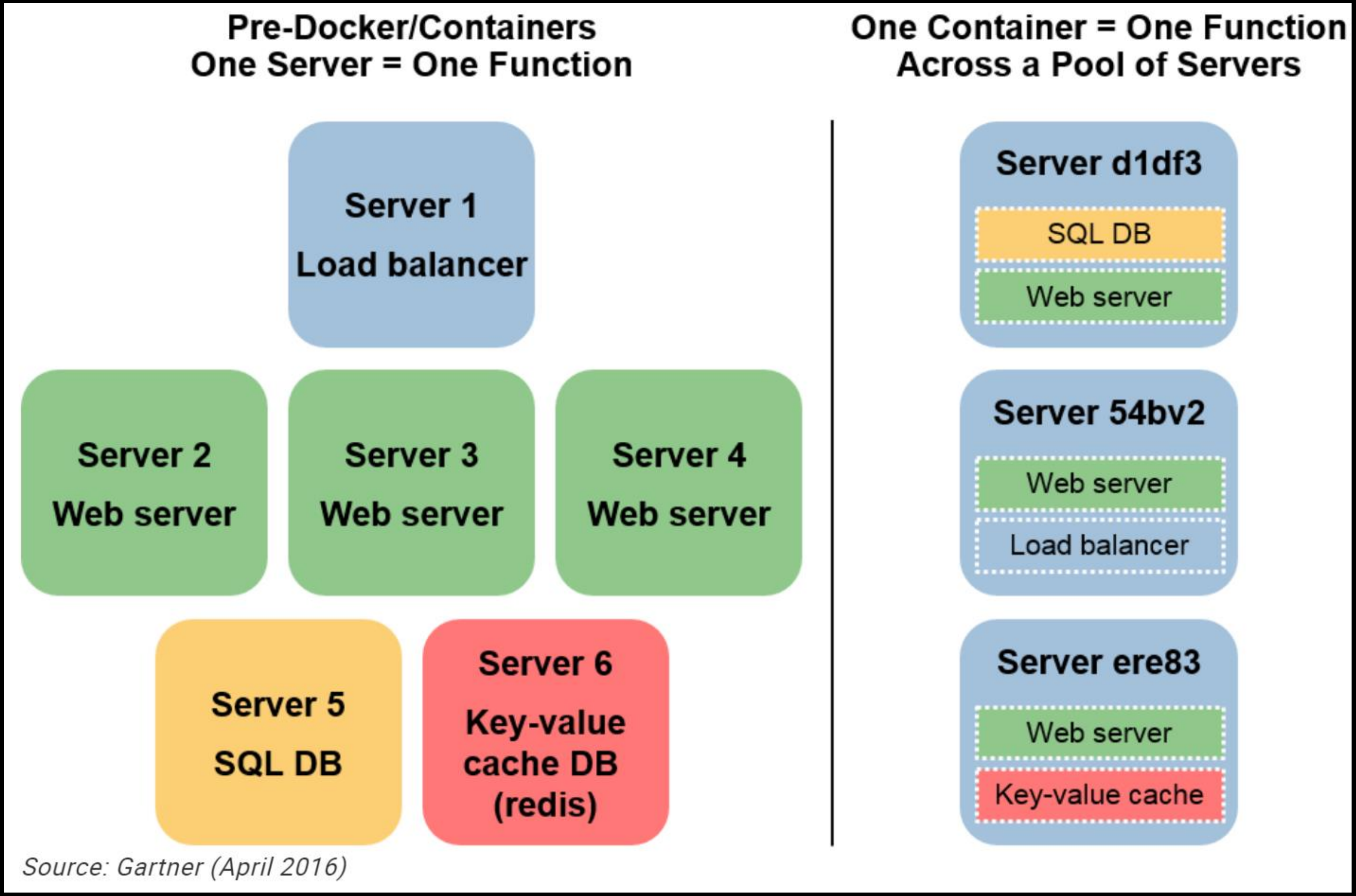
- When a developer checks in code from their laptop, then steps to deployment on AWS are automated



Evolution of Computing

Development Process	Application Architecture	Deployment and Packaging	Application Infrastructure
Waterfall	Monolithic	Physical Server	Datacenter
Agile	N-Tier	Virtual Servers	Hosted
DevOps  Plan Code Build Test Release Deploy Operate Monitor	Microservices	Containers	Cloud

Containers for micro-services



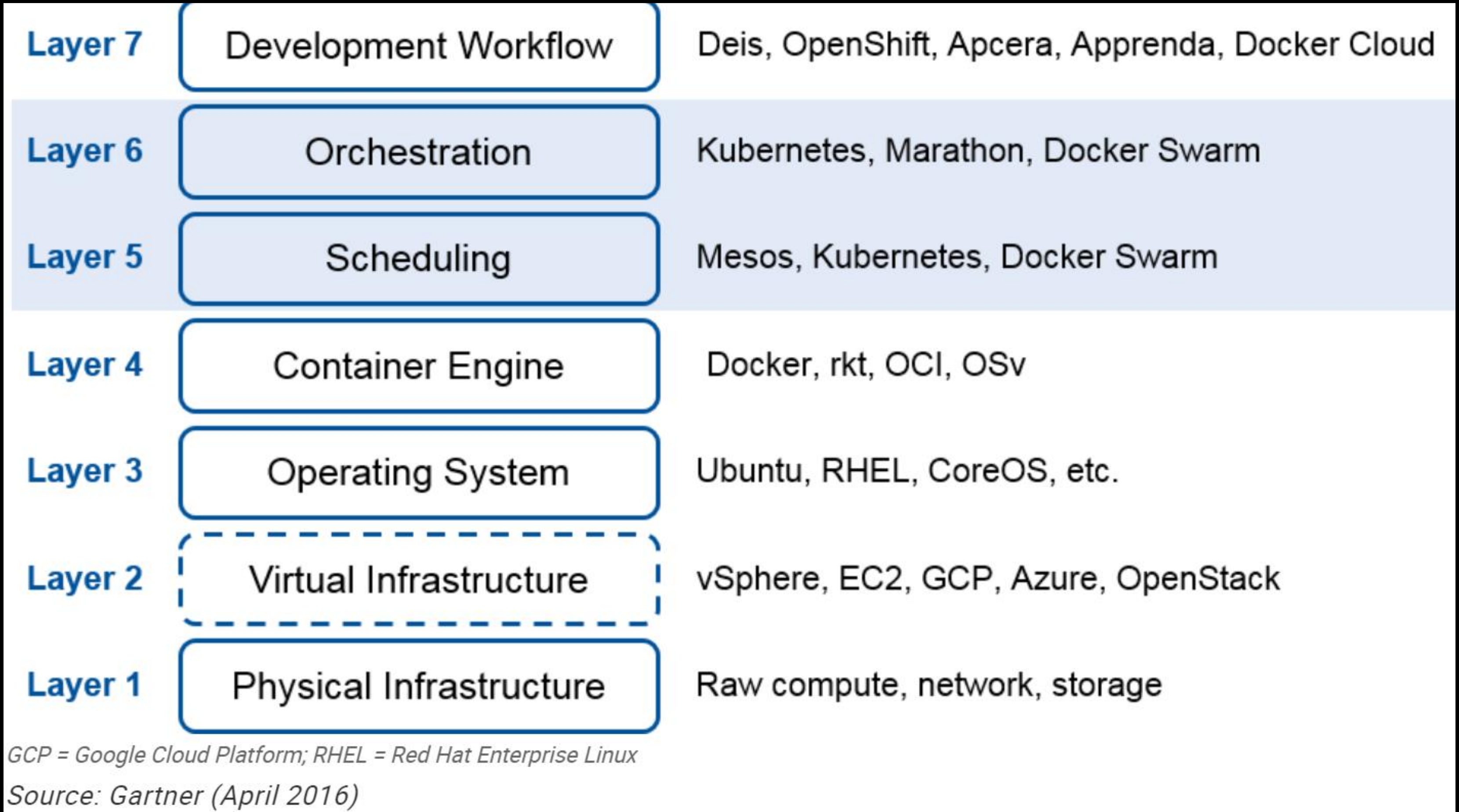
Source: Gartner (April 2016)

Microservices

- **Microservices** are a software development technique—a variant of the service-oriented architecture (SOA) architectural style that structures an application as a collection of loosely coupled services.
  - services are fine-grained and the protocols are lightweight.
  - The benefit of decomposing an application into different smaller services is that it improves modularity.
  - This makes the application easier to understand, develop, test, and become more resilient to architecture erosion.[1] I
  - t parallelizes development by enabling small autonomous teams to develop, deploy and scale their respective services independently.[2]
  - It also allows the architecture of an individual service to emerge through continuous refactoring.[3]
  - Microservices-based architectures enable continuous delivery and deployment.[4][1][5]

https://en.wikipedia.org/wiki/Microservices

Layers of instructor for containers

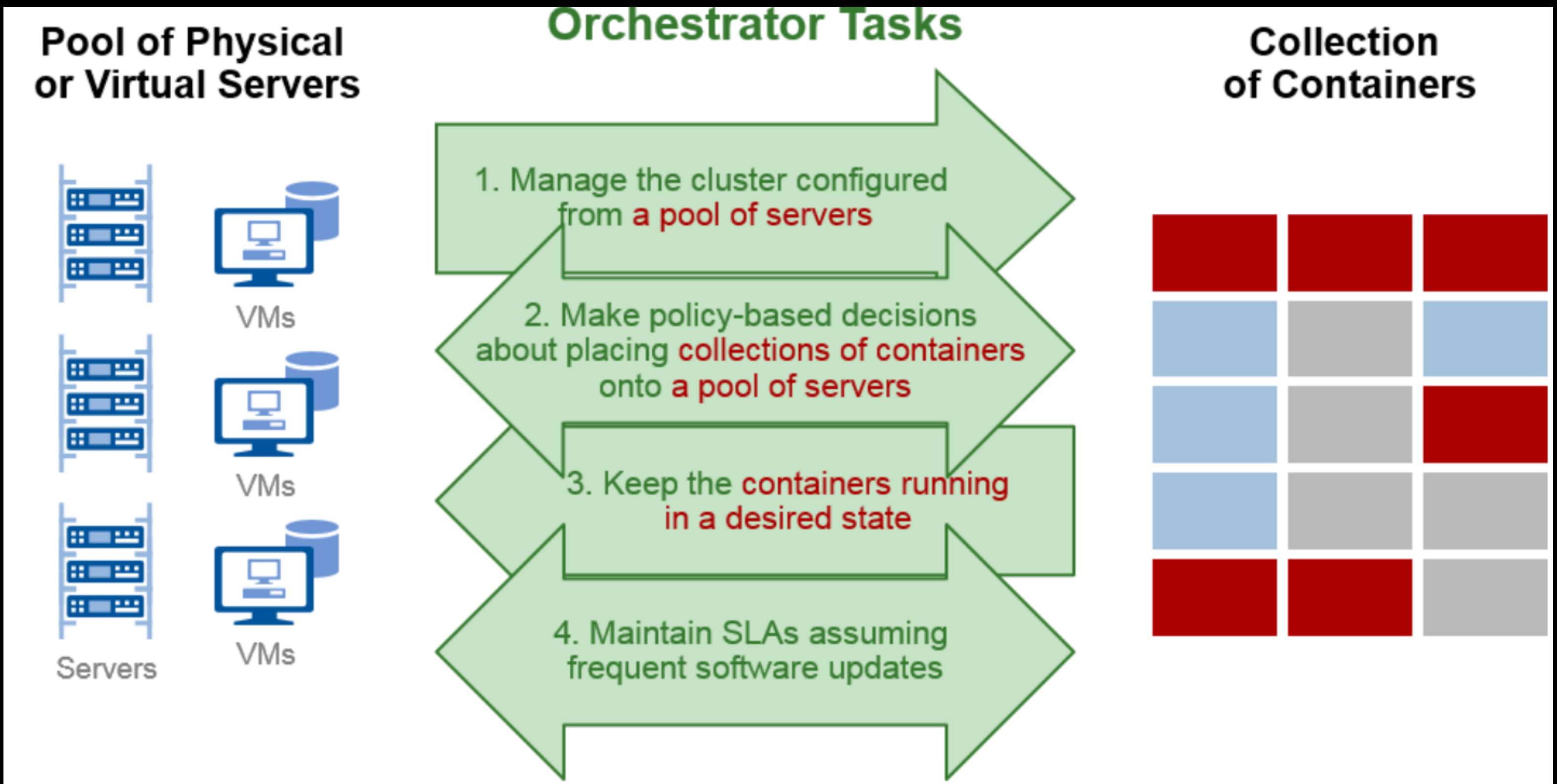


Layer 7	Development Workflow	Deis, OpenShift, Apcera, Apprenda, Docker Cloud
Layer 6	Orchestration	Kubernetes, Marathon, Docker Swarm
Layer 5	Scheduling	Mesos, Kubernetes, Docker Swarm
Layer 4	Container Engine	Docker, rkt, OCI, OSv
Layer 3	Operating System	Ubuntu, RHEL, CoreOS, etc.
Layer 2	Virtual Infrastructure	vSphere, EC2, GCP, Azure, OpenStack
Layer 1	Physical Infrastructure	Raw compute, network, storage

GCP = Google Cloud Platform; RHEL = Red Hat Enterprise Linux

Source: Gartner (April 2016)

Managing containers (mesos)



1. Manage the cluster configured from a pool of servers

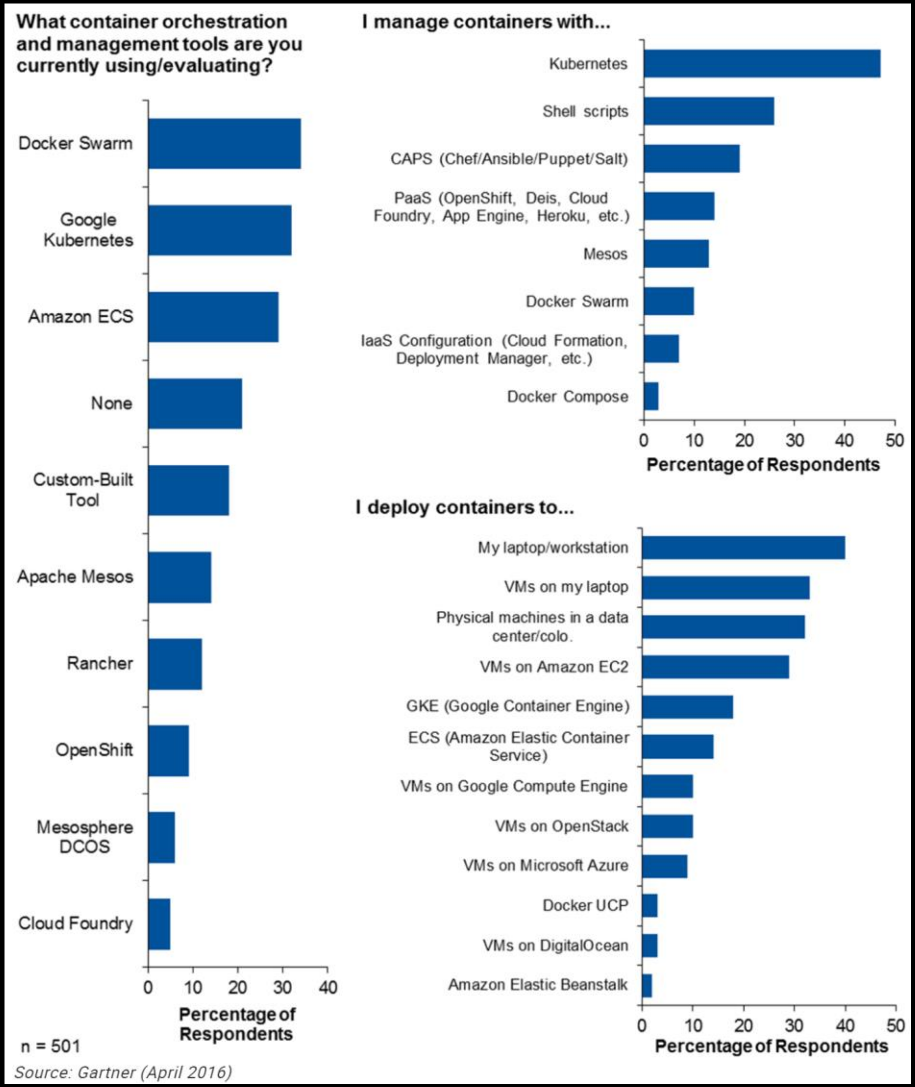
2. Make policy-based decisions about placing collections of containers onto a pool of servers

3. Keep the containers running in a desired state

4. Maintain SLAs assuming frequent software updates

Source: Gartner (April 2016)

Managing containers (survey)



What container orchestration and management tools are you currently using/evaluating?

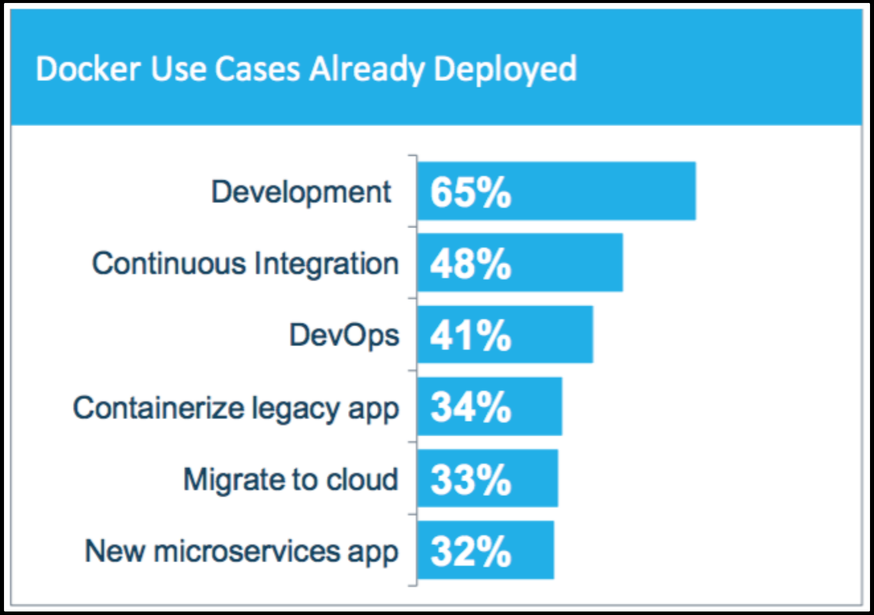
n = 501

I manage containers with...

I deploy containers to...

Source: Gartner (April 2016)

How Containers are Being Used – Survey Says:



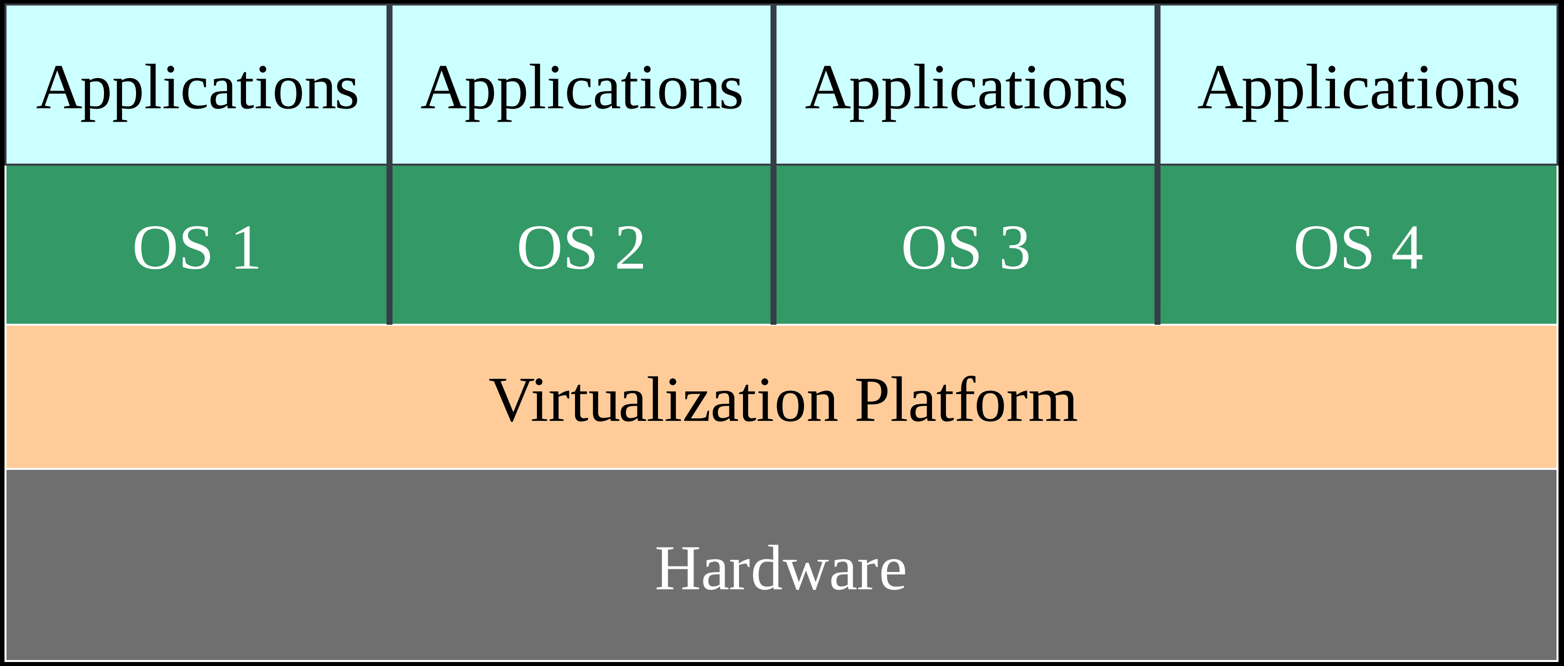
SOURCE: THE EVOLUTION OF THE MODERN SOFTWARE SUPPLY CHAIN, DOCKER SURVEY 2016

- Developer productivity a top use case today
- Building out CI/CD pipelines
  - Consistent container image moves through pipeline
  - Preventing "it worked in dev" syndrome
- Application modernization and portability are also key adoption drivers

Virtualization

Operating System Virtualization

A platform that emulates a hardware platform and allows multiple instances of an OS to use that platform as though they have full and exclusive access to the underlying hardware



Applications | Applications | Applications | Applications

OS 1 | OS 2 | OS 3 | OS 4

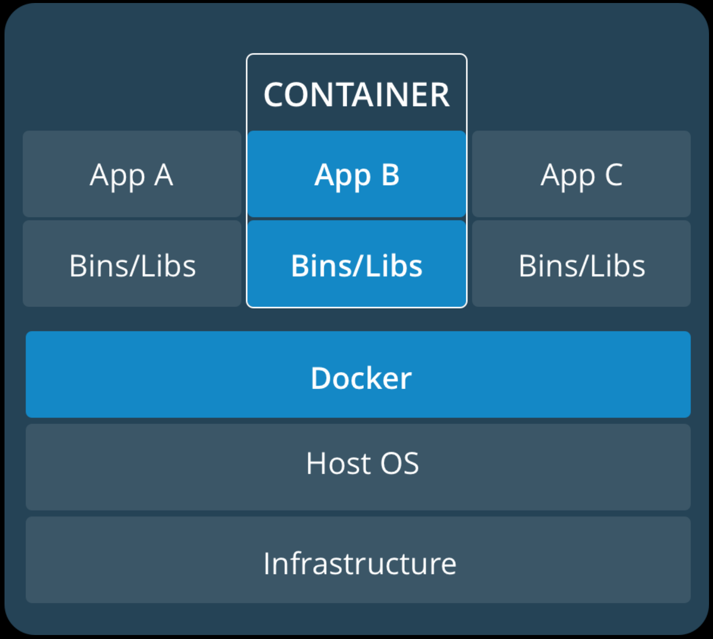
Virtualization Platform

Hardware

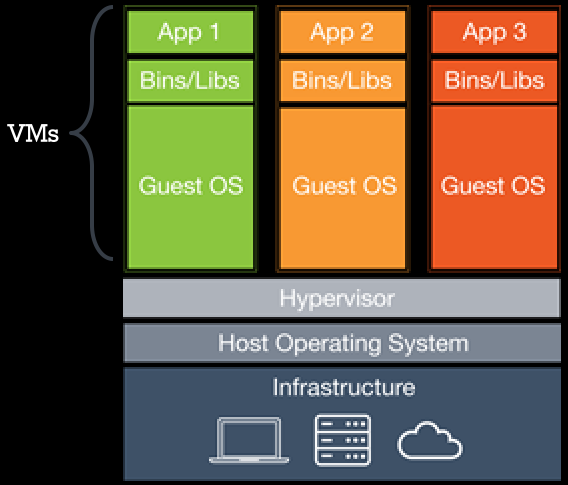
Container

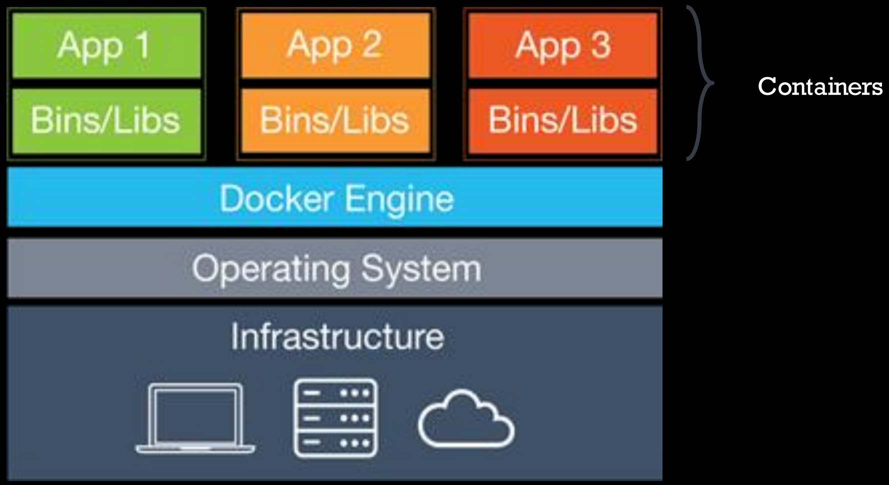
Container

- a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another.



Virtual Machines vs. Containers





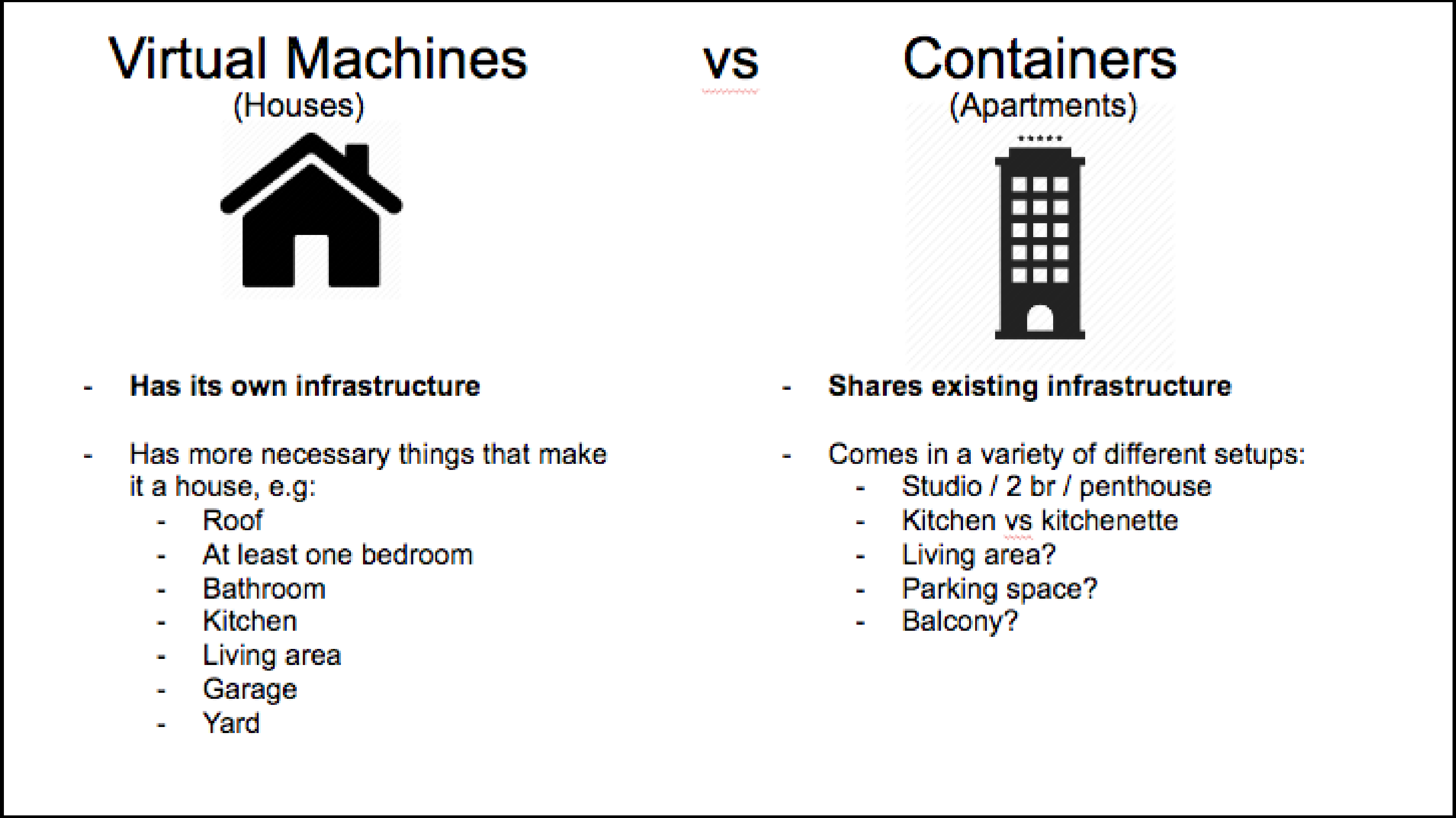
Virtual Machines

- Each virtual machine (VM) includes the app, the necessary binaries and libraries and an **entire guest operating system**

Containers

- Containers include the app & all of its dependencies, but **share the kernel** with other containers.
- Run as an isolated process in userspace on the host OS
- Not tied to any specific infrastructure – containers run on any computer, infrastructure and cloud.

Analogy: VM's to Containers ….
Houses to apartments



Virtual Machines (Houses) vs Containers (Apartments)

Virtual Machines

(Houses)

- **Has its own infrastructure**

- Has more necessary things that make it a house, e.g:
  - Roof
  - At least one bedroom
  - Bathroom
  - Kitchen
  - Living area
  - Garage
  - Yard

Containers

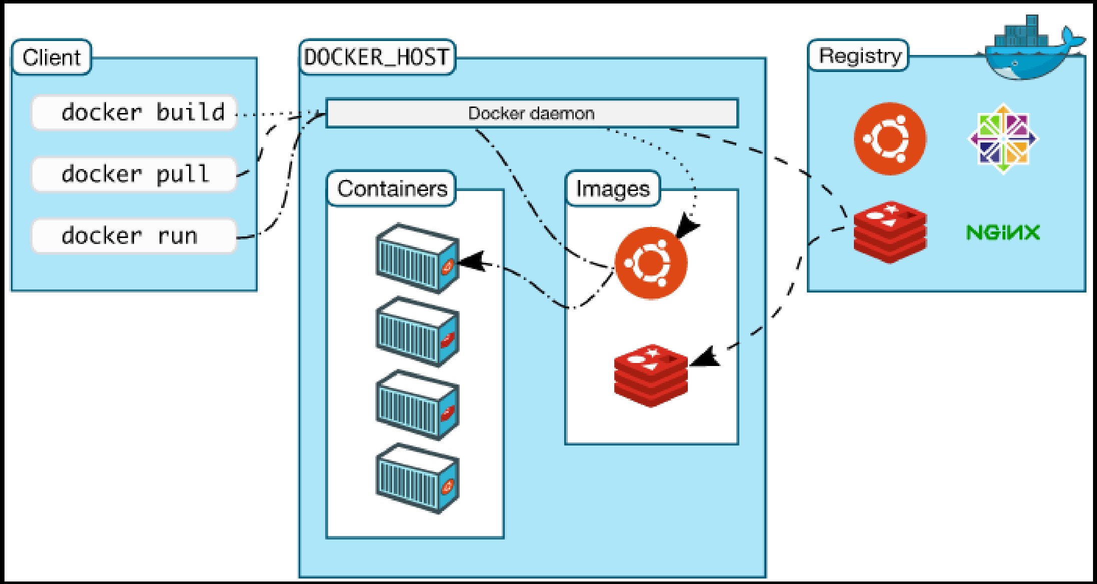
(Apartments)

- **Shares existing infrastructure**

- Comes in a variety of different setups:
  - Studio / 2 br / penthouse
  - Kitchen vs kitchenette
  - Living area?
  - Parking space?
  - Balcony?

Docker Architecture

- Docker client – Command Line Interface (CLI) for interfacing with the Docker
- Dockerfile – Text file of Docker instructions used to assemble a Docker Image
- Image – Hierarchies of files built from a Dockerfile, the file used as input to the docker build command
- Container – Running instance of an Image using the docker run command
- Registry – Image repository



Source: Docker docs and https://docs.docker.com/glossary/

Docker terminology

Run Hello World in a container

- docker run ubuntu echo "Hello World"
- View Docker information
  - docker images [-a]
  - docker ps –a

Docker image

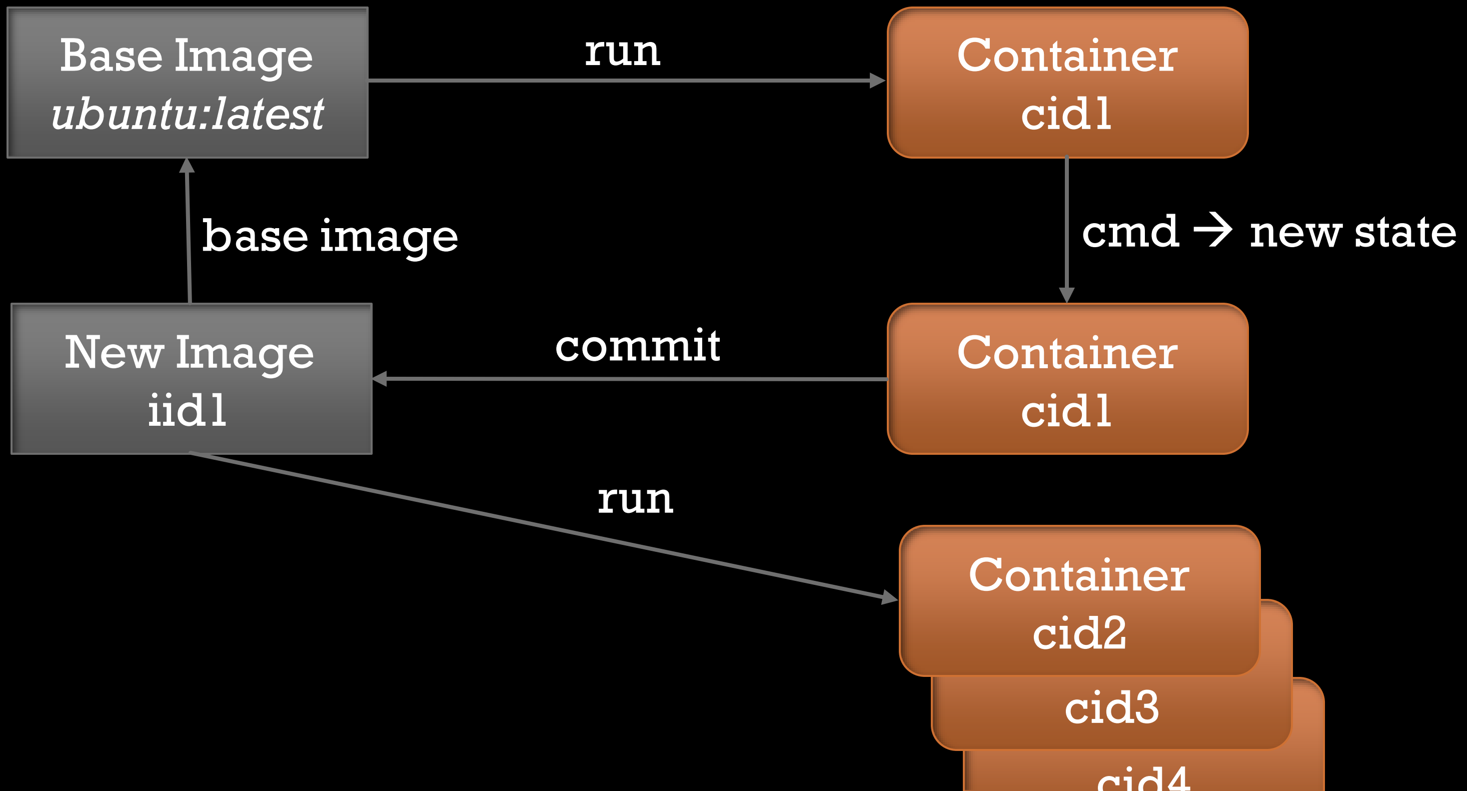
- Persisted snapshot that can be run
- docker image commands
  - images: List all local images
  - run: Create a container from an image and execute a command in it
  - tag: Tag an image
  - pull: Download image from repository
  - rmi: Delete a local image
    - This will also remove intermediate images if no longer used

Container

- Runnable instance of an image
- docker container commands
  - ps: List all running containers
  - ps –a: List all containers (incl. stopped)
  - top: Display processes of a container
  - start: Start a stopped container
  - stop: Stop a running container
  - pause: Pause all processes within a container
  - rm: Delete a container
  - commit: Create an image from a container

Image vs. Container

- Image is the specification and container is the running instance
- Container is created as the result of run image
- An image can be created by committing a container



Dockerfile

- Simple text file
- Using docker build on the Dockerfile, an image is created
- Can be versioned e.g., Git or SVN
- Docker Hub can automatically build images based on dockerfiles on Github

Dockerfile Example

- Dockerfile:

  -
    ```
    FROM ubuntu
    ADD dir /files
    CMD ["bash","someScript"]
    ```
- docker build [DockerFileDir]
- docker inspect [imageId]

Mount Volumes

- Host files can be mounted (included) in the running container
- docker run –ti –v /hostLog:/log ubuntu
- Run second container: Volume can be shared
  - docker run –ti --volumes-from firstContainerName ubuntu
- Volumes are used to shared and persisted data, which can be accessed by many Docker containers

Docker Hub

- Public repository of Docker images
  - https://hub.docker.com/
  - docker search [term]
- Automated: Has been automatically built from Dockerfile
  - Source for build is available on GitHub

Summary of Topics

- Why Docker containers
  - Simplifies hardware and operating system management
- Virtualization
  - Clear separation of hardware from software
    - E.g., Allows running Windows on a Mac computer
- Container defined
  - Run an app and its operating system in a process on any computer
- Docker terminology
  - Image, container, dockerfile