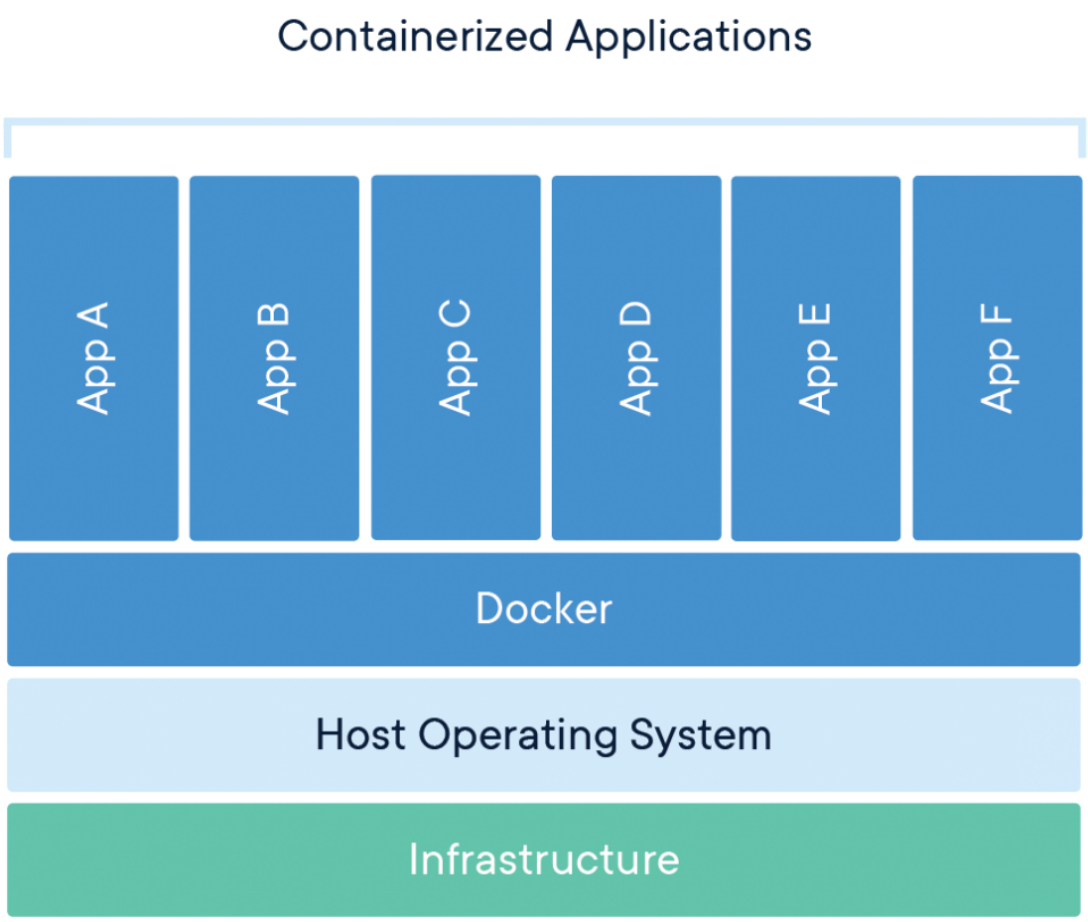
Containerization



- *See Virtual-Machine infographic (image) for contrast*

Preface

Terminologies like **"serverless"** are often used in replacement of **"containerization"**; however, *always*, **these two labels are two the very same**.

- See *What is AWS Lambda*
  - (`serverless.com` *is an AWS-Managed, Open-Source PaaS/Framework*)
    - https://www.serverless.com/aws-lambda

Overview

> **Containerization** is the *packaging of software code* with just the operating system (OS) libraries and *dependencies required* to run the code to create a single lightweight executable—called a container that runs consistently on *any infrastructure*.

> *More portable and resource-efficient than virtual machines* (VMs), containers have become the de facto compute units of **modern cloud-native applications**.

- **IBM**, *Containerization Explained*

Statistics

Containers in The Enterprise

Productivity

74% Report Increased Productivity -- up from 48% in 2017.

76% Report Faster Time-to-Market -- up from 45% in 2017.

Security

73% of 2020 respondents reported improved company security — up from 39% in 2017.

Benefits

Containerization offers significant benefits to developers and development teams. Among these are the following:

- **Portability**: A container creates an executable package of software that is abstracted away from (not tied to or dependent upon) the host operating system, and hence, is portable and able to run uniformly and consistently across any platform or cloud.
- **Agility**: The open source Docker Engine for running containers started the industry standard for containers with simple developer tools and a universal packaging approach that works on both Linux and Windows operating systems. The container ecosystem has shifted to engines managed by the Open Container Initiative (OCI). Software developers can continue using agile or DevOps tools and processes for rapid application development and enhancement.
- **Speed**: Containers are often referred to as "lightweight," meaning they share the machine's operating system (OS) kernel and are not bogged down with this extra overhead. Not only does this drive higher server efficiencies, it also reduces server and licensing costs while speeding up start-times as there is no operating system to boot.
- **Fault isolation**: Each containerized application is isolated and operates independently of others. The failure of one container does not affect the continued operation of any other containers. Development teams can identify and correct any technical issues within one container without any downtime in other containers. Also, the container engine can leverage any OS security isolation techniques—such as SELinux access control—to isolate faults within containers.
- **Efficiency**: Software running in containerized environments shares the machine's OS kernel, and application layers within a container can be shared across containers. Thus, containers are inherently smaller in capacity than a VM and require less start-up time, allowing far more containers to run on the same compute capacity as a single VM. This drives higher server efficiencies, reducing server and licensing costs.

- **Ease of management**: A container orchestration platform automates the installation, scaling, and management of containerized workloads and services. Container orchestration platforms can ease management tasks such as scaling containerized apps, rolling out new versions of apps, and providing monitoring, logging and debugging, among other functions. Kubernetes, perhaps the most popular container orchestration system available, is an open source technology (originally open-sourced by Google, based on their internal project called Borg) that automates Linux container functions originally. Kubernetes works with many container engines, such as Docker, but it also works with any container system that conforms to the Open Container Initiative (OCI) standards for container image formats and runtimes.
- **Security**: The isolation of applications as containers inherently prevents the invasion of malicious code from affecting other containers or the host system. Additionally, security permissions can be defined to automatically block unwanted components from entering containers or limit communications with unnecessary resources.