Containers- Overview

KOENIG

step forward

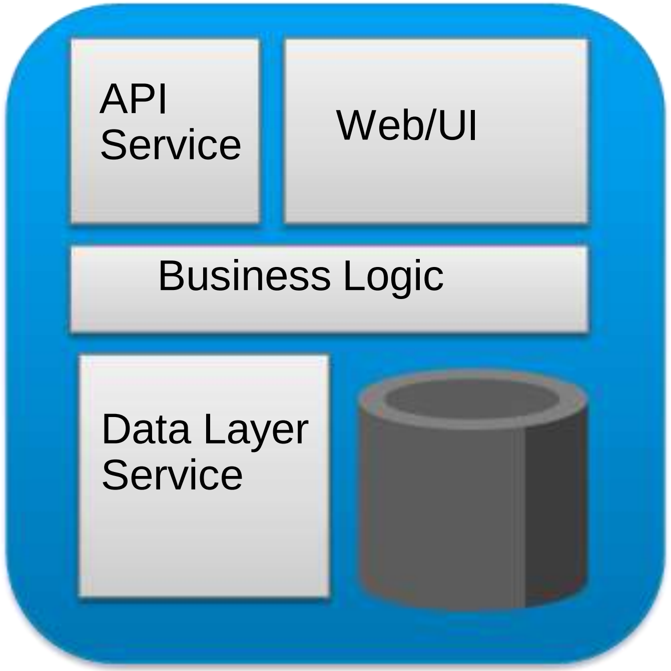
Cloud-Native Principles

Cloud-native principles apply to containers:

- Isolated unit of work that does not require OS dependencies
- Actively scheduled and managed by an orchestration process
- Loosely coupled from any dependencies
- Use microservices

Containers and Microservices

The characteristics of containers (lightweight, easily packaged, can run anywhere) align with the goals of the microservices architecture.



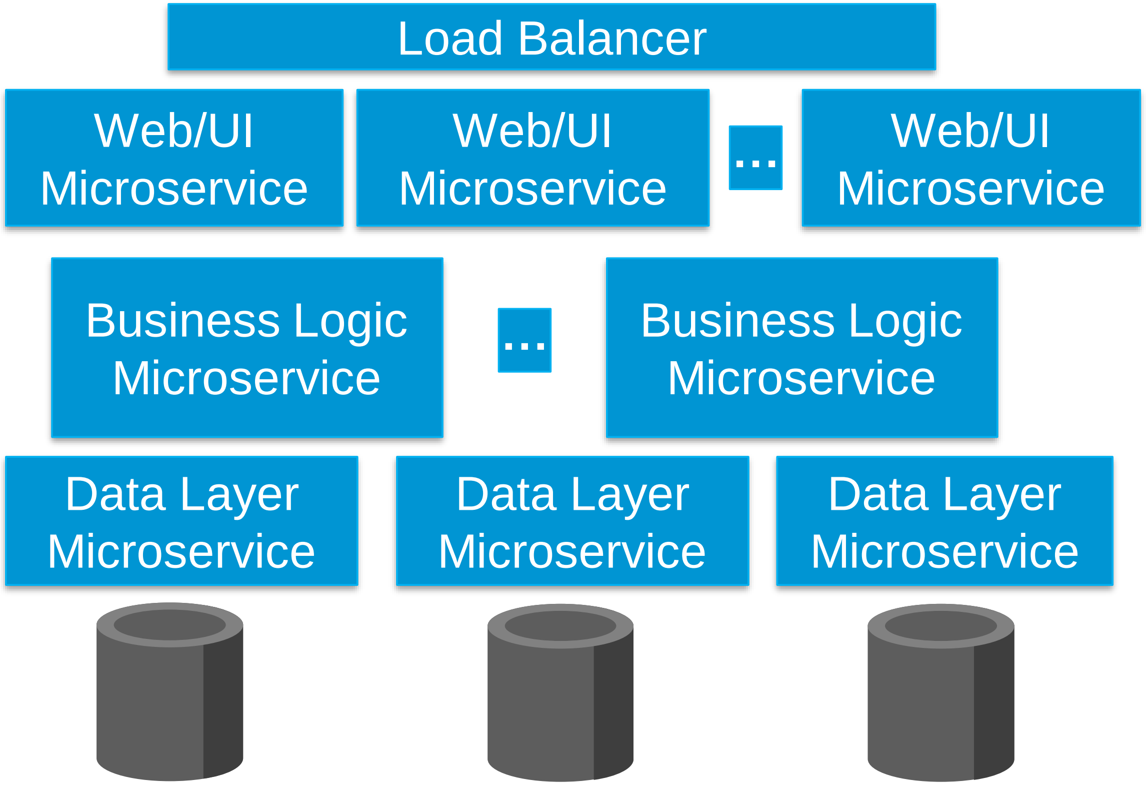
API Service

Web/UI

Business Logic

Data Layer Service

Monolithic Application



Load Balancer

Web/UI Microservice

Web/UI Microservice

...

Web/UI Microservice

Business Logic Microservice

...

Business Logic Microservice

Data Layer Microservice

Data Layer Microservice

Data Layer Microservice

Application as Microservices

Container Benefits

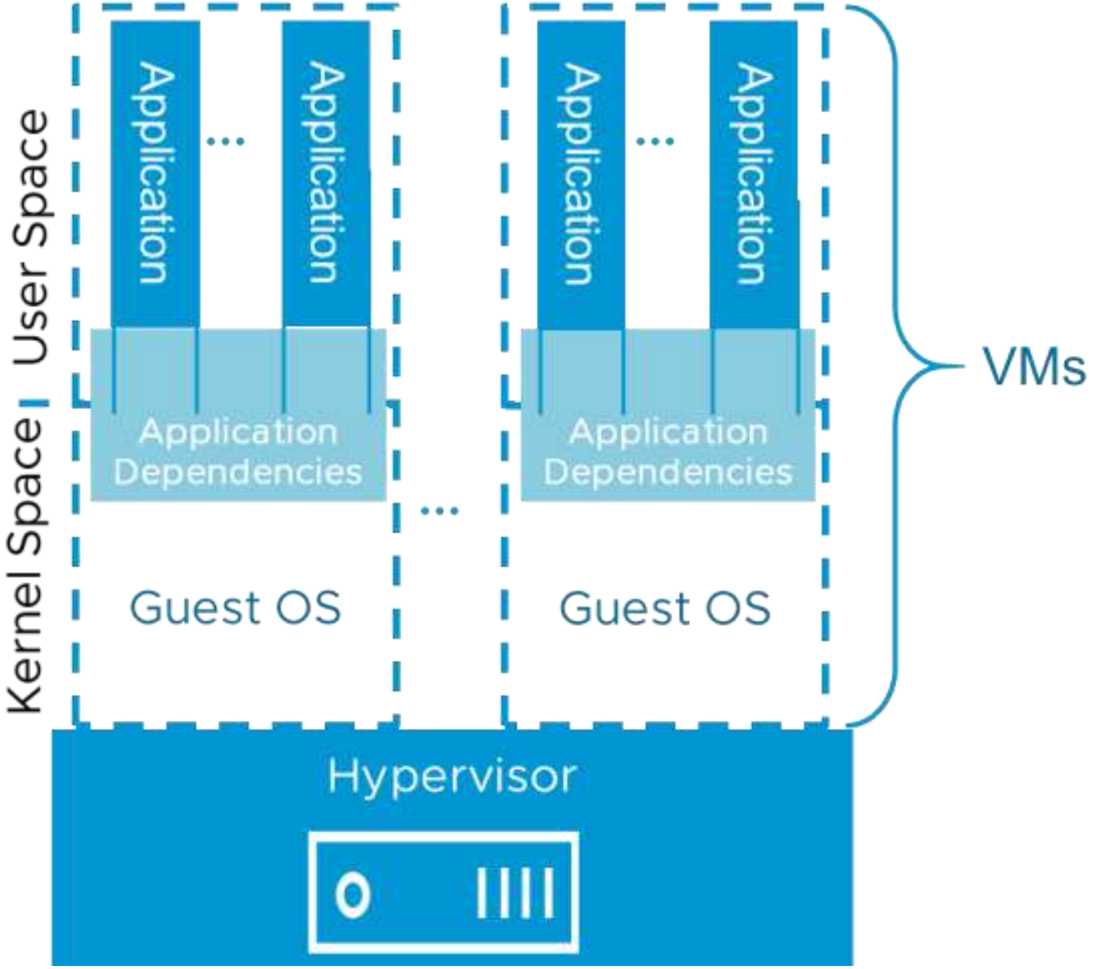
Containers provide the following benefits:

- Velocity
- Portability
- Reliability
- Efficiency
- Self-service
- Isolation

Applications in Virtual Machines at Runtime

Virtual machines encapsulate a full OS with the following components:

- Running processes (such as applications)
- Memory management
- Device drivers
- Daemons
- Libraries

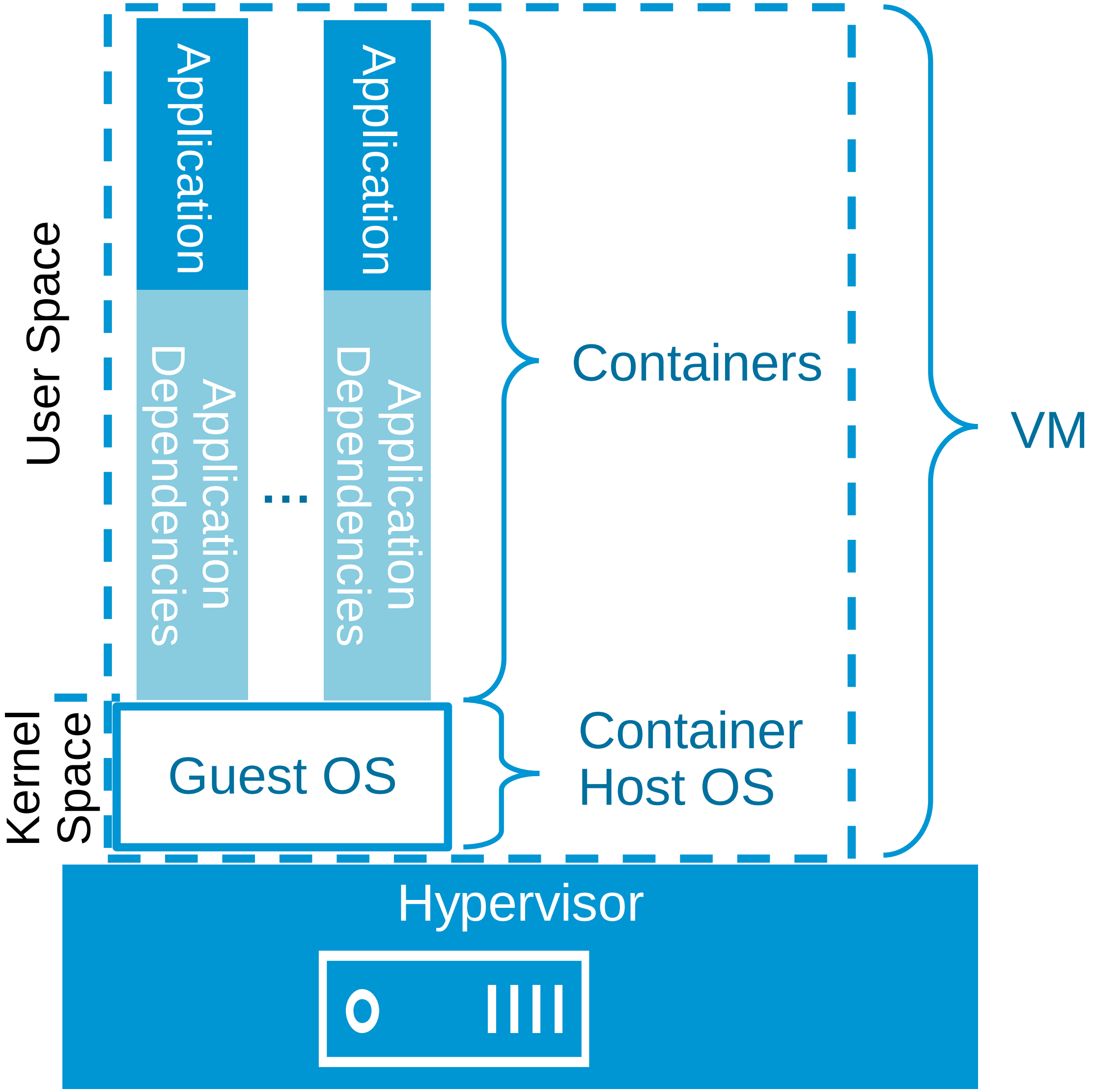


Applications in Containers at Runtime

Containers are the encapsulation of an application process with the application dependencies.

Containers are ultraportable. A container can run on any container host with the same operating system kernel that is specified by that container.

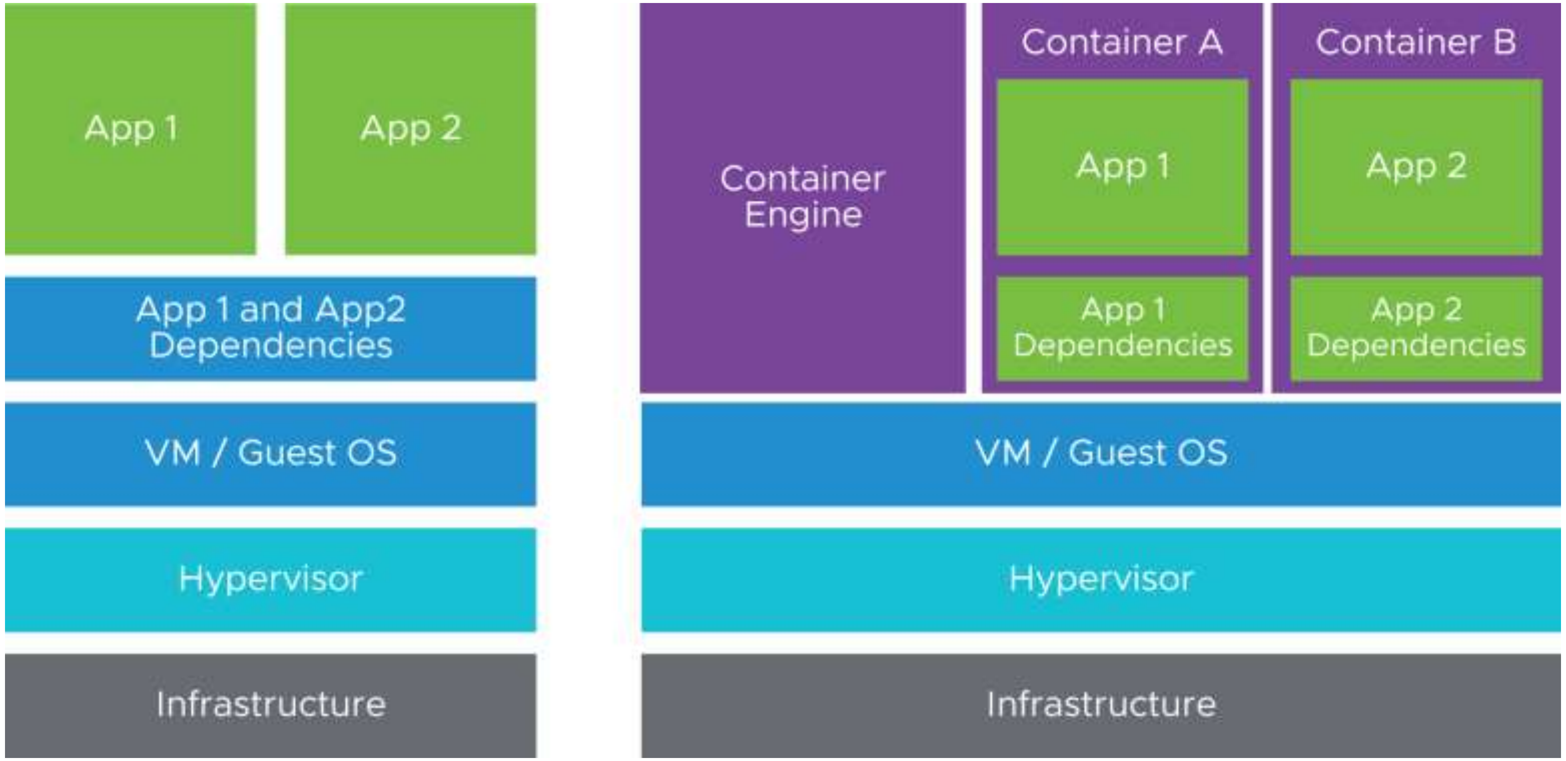
The word Docker is often used as a synonym for many aspects of container technologies.



Virtual Machines and Containers (1)

Each VM provides virtual hardware that the guest OS uses to execute applications. Multiple applications run on a single physical server while still being logically separated and isolated.

With containers, developers take a streamlined base OS file system and layer on only the required binaries and libraries on which the application depends.



Virtual Machines and Containers (2)

Compare the characteristics of virtual machines and containers.

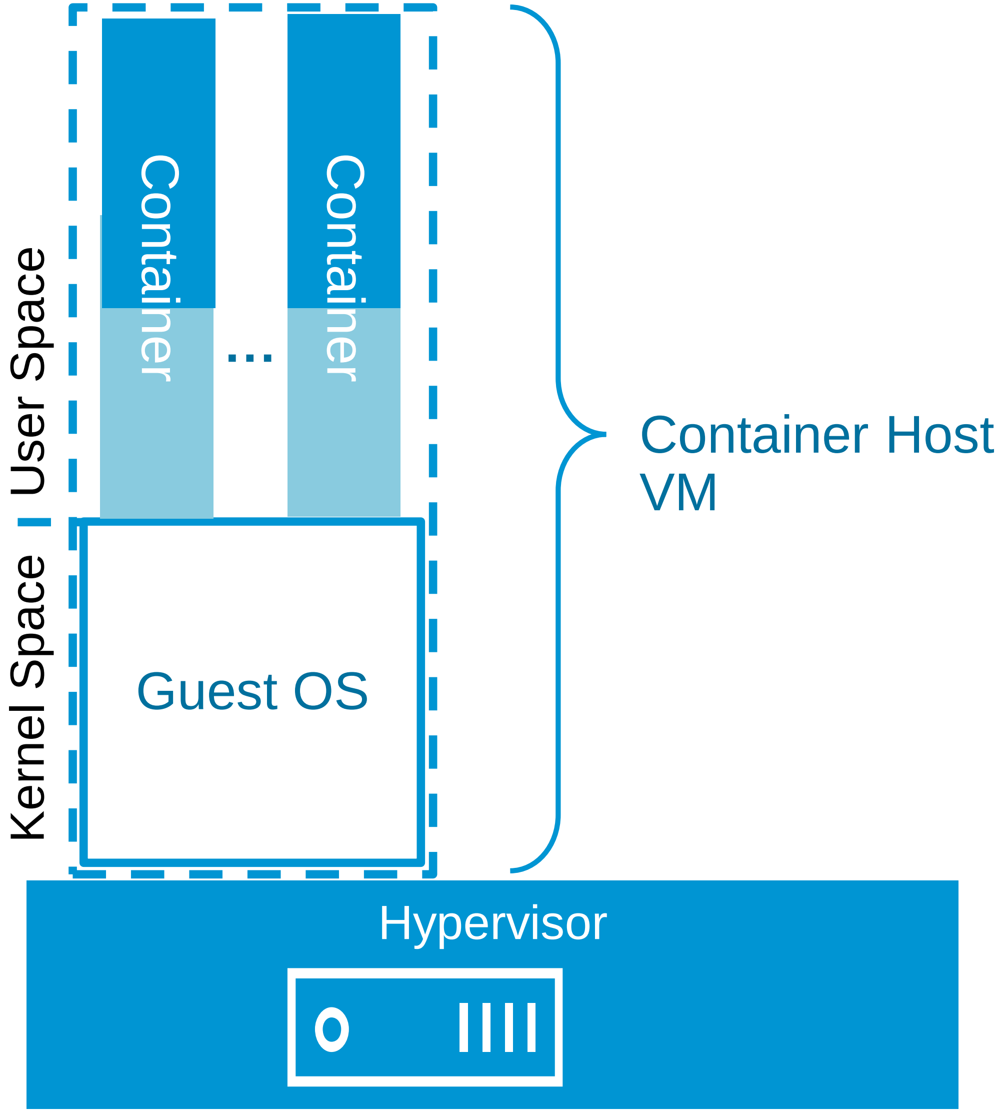
Virtual Machines	Containers
Encapsulation of an entire operating system.	Encapsulation of an application and dependent binaries or libraries.
Scheduled by the hypervisor.	Scheduled by the container scheduler.
Run on the hypervisor.	Run on the host server.
Starting a VM means starting the virtual hardware and the operating system (seconds to minutes).	Starting a container means starting the application process (milliseconds to seconds).

Container Hosts

The container host runs the operating system on which the containers run.

Using virtual machines as container hosts has many advantages:

- Flexibility
- Scalability
- Security



About Container Hosts

Container hosts can be of the following types:

- Standard OS with a container engine installed:
  - Ubuntu with Docker or another container engine

- OS that is developed specifically with containers in mind:
  - Photon
  - CoreOS

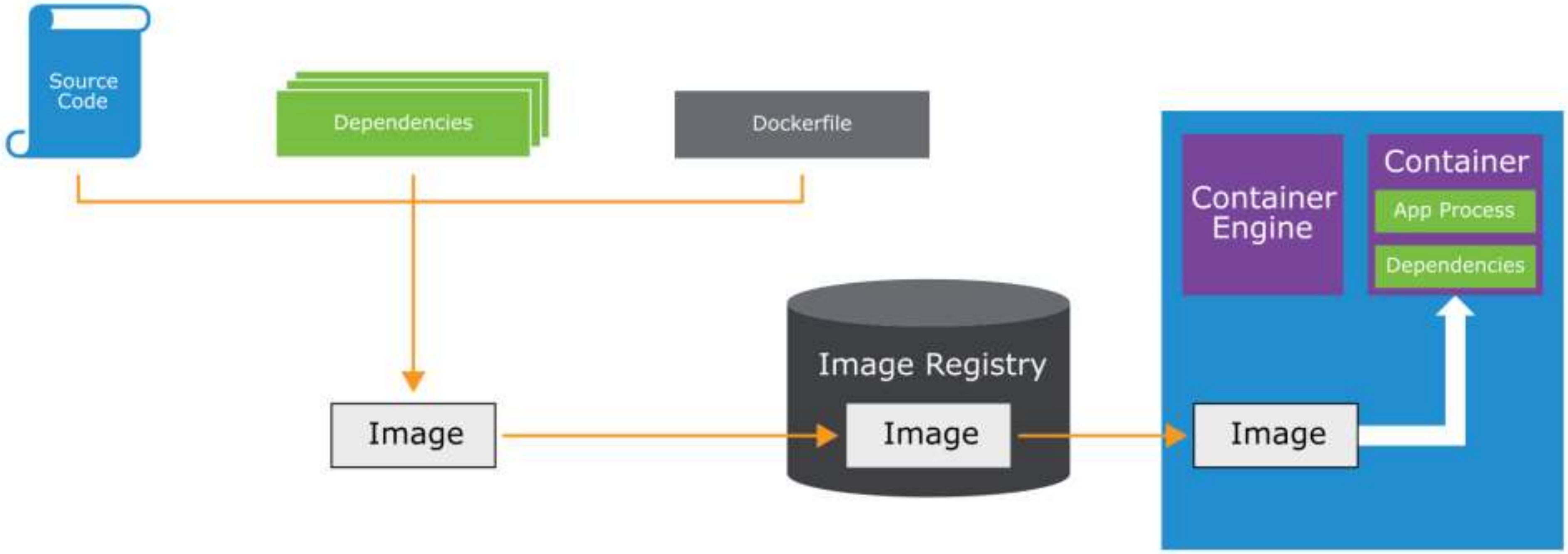
Container hosts can be a virtual machine or a physical machine (bare metal):

- Using VMs has many benefits, such as easy management and scalability.

Typical Container Workflow

A container workflow includes these steps:

1. Build an image from the source code and dependencies.
2. Send the image to the image registry.
3. Take the image from the image registry.
4. Run the image as a container.

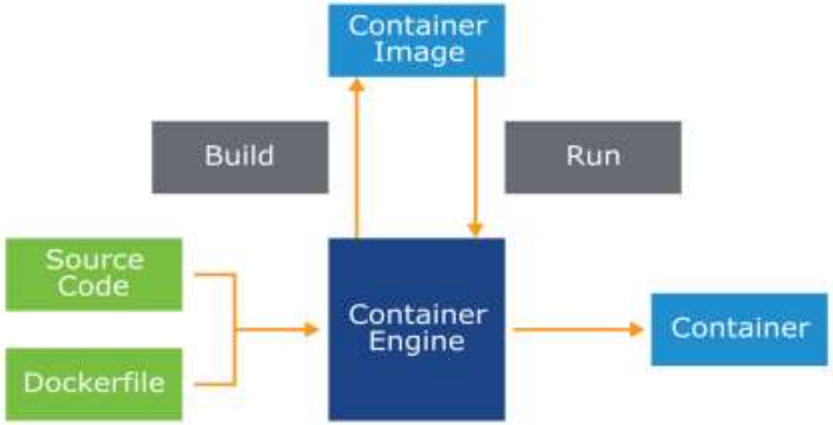


About Container Engines

A container engine is a control plane installed on each container host. It manages the containers on that host.

Container engines work in the following ways:

- Build container images from the source code (for example, Dockerfile).
- Alternatively, start container images from a repository.
- Create running containers based on a container image.
- Commit a running container to an image.
- Save an image and push it to a repository.
- Stop and remove containers.
- Suspend and restart containers.
- Report container status.



The Dockerfile and Programmatic Construction

VM creation is not guaranteed to be scripted or reproducible.

Dockerfile is a plaintext file that clearly defines each stage that is required to build a container image.

Dockerfile produces a consistent and reproducible container image

You run the `docker build` command to create an image from Dockerfile.

You run the `docker run` command to create and start a container.

```
FROM ubuntu:18.04
RUN apt-get update && apt-get install -y php=8.0.8 nginx=1.20.1
COPY ./index.php /var/www/html/index.php
EXPOSE 80
CMD ["/var/www/html/index.php", "-D", "FOREGROUND"]
```

Container Images

An image is effectively a stopped container that is loaded on the container host. An image is like a VM snapshot.

Images are stored to and retrieved from an image repository server.

They contain application code and application dependencies:

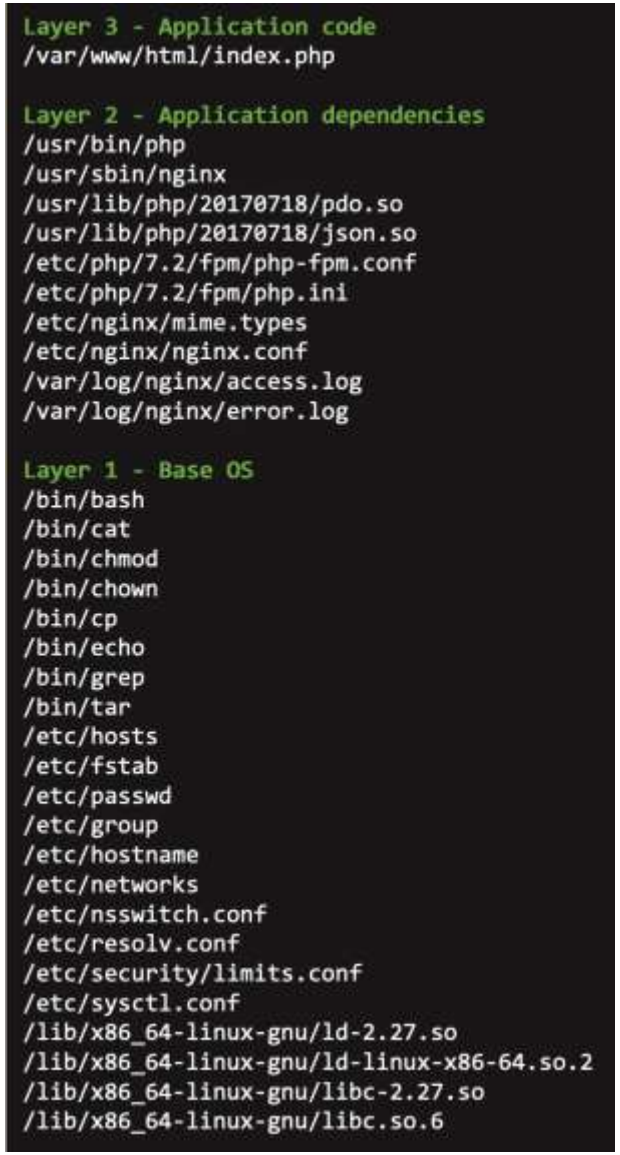
- Do not contain kernel or device drivers.
- Are built using a layered file system, for example:

– Layer 1: Base OS layer, including typically low-level OS libraries.

Each subsequent layer stores additional components as defined in the Dockerfile.

– Layer 2: Application-specific dependencies, such as NGINX web server and PHP runtime.

– Layer 3: Application code, such as index.php.



```
Layer 3 - Application code
/var/www/html/index.php

Layer 2 - Application dependencies
/usr/bin/php
/usr/sbin/nginx
/usr/lib/php/20170718/pdo.so
/usr/lib/php/20170718/json.so
/etc/php/7.2/fpm/php-fpm.conf
/etc/php/7.2/fpm/php.ini
/etc/nginx/mime.types
/etc/nginx/nginx.conf
/var/log/nginx/access.log
/var/log/nginx/error.log

Layer 1 - Base OS
/bin/bash
/bin/cat
/bin/chmod
/bin/chown
/bin/cp
/bin/echo
/bin/grep
/bin/tar
/etc/hosts
/etc/fstab
/etc/passwd
/etc/group
/etc/hostname
/etc/networks
/etc/nsswitch.conf
/etc/resolv.conf
/etc/security/limits.conf
/etc/sysctl.conf
/lib/x86_64-linux-gnu/ld-2.27.so
/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
/lib/x86_64-linux-gnu/libc-2.27.so
/lib/x86_64-linux-gnu/libc.so.6
```

Images and Containers

An image is the result of a build.

A container is a running instance from an image.

```
$ docker images
php              latest      8c811b4aec35        2 weeks ago         1.15 MB
mysql            latest      a8a59477268d        4 weeks ago         445 MB
foo.com/mysql    latest      a8a59477268d        4 weeks ago         445 MB

$ docker ps
CONTAINER ID   IMAGE   COMMAND    CREATED              STATUS              PORTS      NAMES
Acc5c8f58392   mysql   “mysqld”   About a minute ago   Up About a minute   3306/tcp   myWebsiteDB
```

Starting and Stopping Containers

You run the following commands to start and stop containers:

- `docker run`: Starts a new container from an image
- `docker stop`: Stops a running container
- `docker rm`:
  - Deletes a container (must be stopped)
  - Add `–f` to both stop and remove a container

Managing Images

The following commands help to manage images:

- `docker images`: Displays a list of images on the machine
- `docker rmi`: Deletes an image
- `docker build`: Builds an image from a Dockerfile
- `docker tag`: Adds tags to an image
- `docker pull` and `docker push`: Pulls and pushes images to and from a registry

Additional Docker Commands

You might run these additional commands when working with images:

- `docker ps`:
  - Retrieves a list of running containers
  - Add `-a` to include non-running containers
- `docker logs`: Displays a container's log output
- `docker exec`:
  - Runs a command within a container
  - Can also start a shell within a container (if available)
- `docker network`: Creates a network

Container Registry

A container registry provides a central location for container image storage.

Types of registries include hosted and self-hosted:

- Hosted: Docker Hub, Google Container Registry
- Self-hosted: Artifactory, Harbor, Quay

Container registries can be public or private.

Images are built on a build host and pushed to a registry.

Project Harbor

Harbor is an open-source enterprise-class registry server. It is integrated into many VMware products.

Harbor includes the following features:

- Identity integration and role-based access control (RBAC)
- Security vulnerability scanning (Clair or Trivy)
- Content trust and image signing (Notary)
- Policy-based image replication
- Helm chart management

Additional References

Title	Location
Container 101 for the vSphere Admin	https://www.youtube.com/watch?v=NeJ20lbzv0c
Google: 'EVERYTHING at Google runs in a container'	https://www.theregister.co.uk/2014/05/23/google_containerization_two_billion/
What is a Container?	https://www.youtube.com/watch?v=EnJ7qX9fkcU
The History of Container Technology	https://linuxacademy.com/blog/containers/history-of-container-technology/
A Brief History of Containers: From the 1970s to 2017	https://blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016
Container vs. Process	https://sites.google.com/site/mytechnicalcollection/cloud-computing/docker/container-vs-process
Container vs. VM	https://sites.google.com/site/mytechnicalcollection/cloud-computing/docker/container-vs-vm