Containers & Kubernetes

Session #01

Containers: Introduction

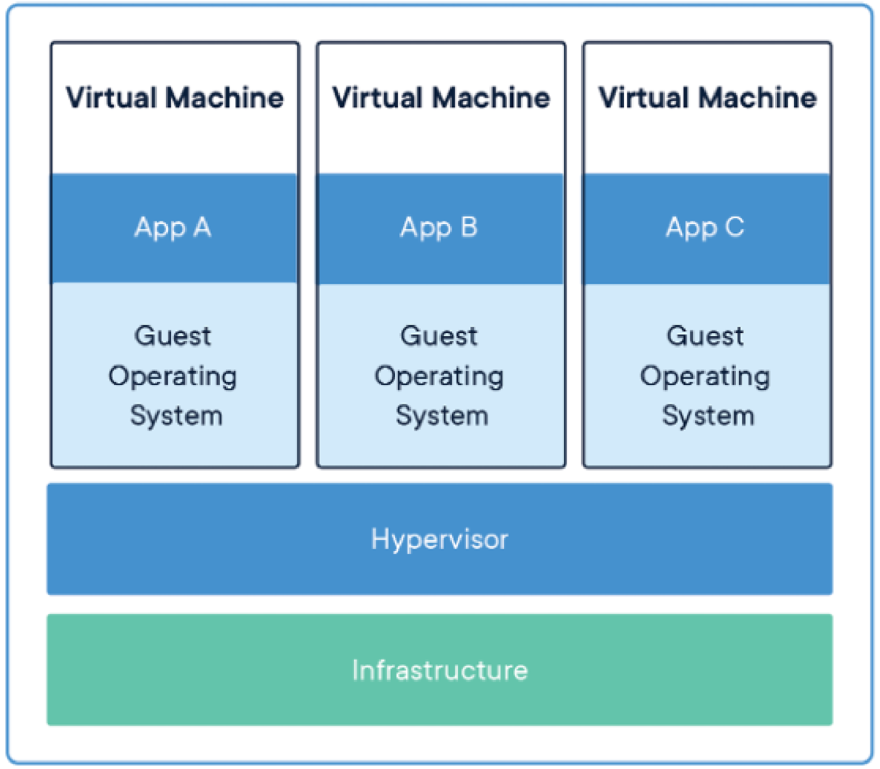
What is a container?

Containers: Introduction

- A method of operation system virtualization
- A way to wrap an application into its own isolated box
- Includes only the binaries needed to support the application
- Isolates an app with its own view of the host from the perspectives of memory, CPU and network

VM vs Containers

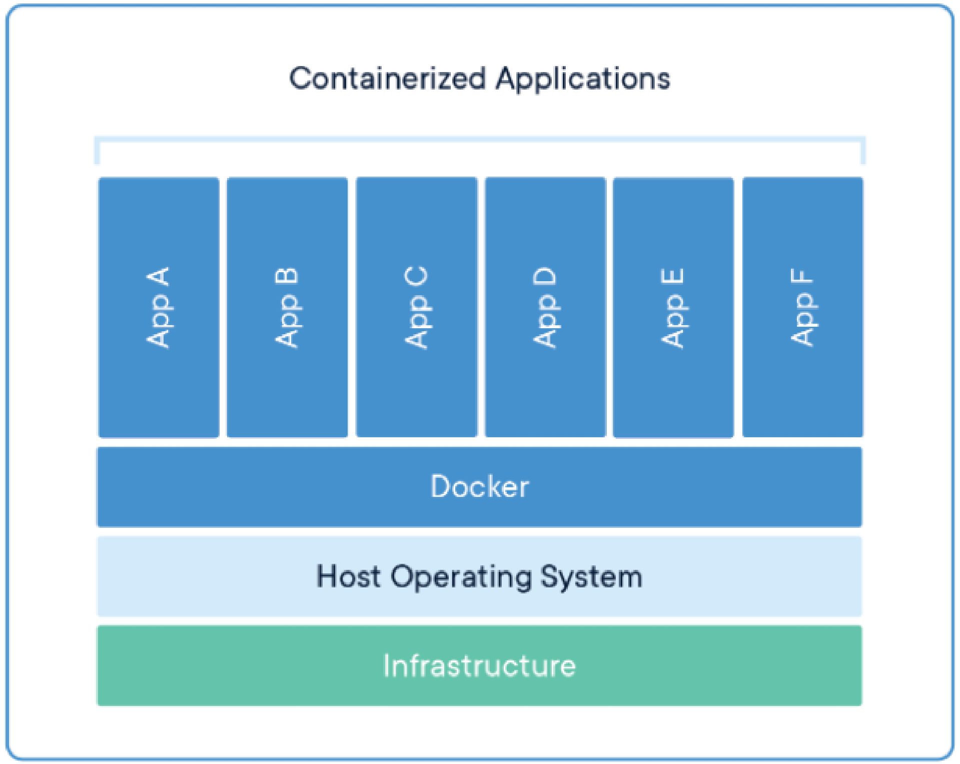
Containers: Introduction



Virtual machines

Virtualize the hardware

VMs as units of scaling



Containers

Virtualize the operating system

Applications as units of scaling

Benefits of using containers

Containers: Introduction

Any OS	Anywhere	Any app	Any language
Linux	On-premises	Monolith	Java
Windows	Cloud	Microservice	.Net
			Python
			Node

Benefits of using containers

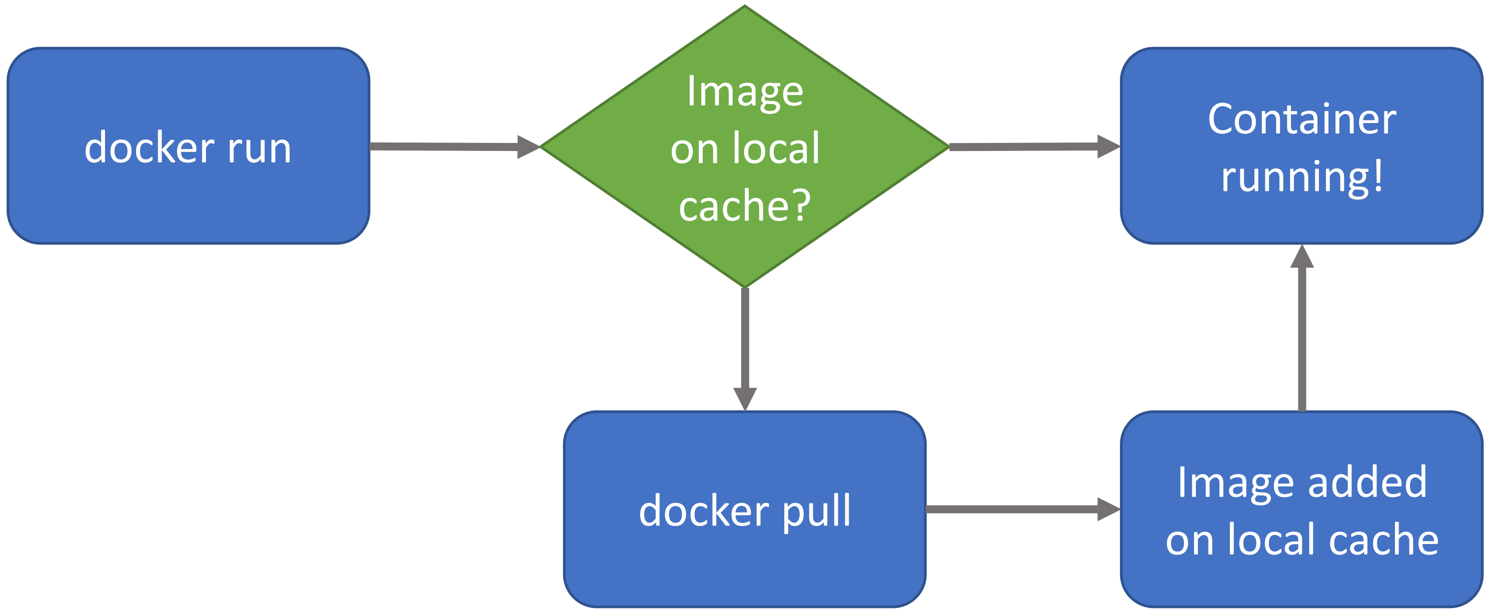
Containers: Introduction

- **Agility**: Ship apps faster
- **Portability**: Easily move workloads
- **Density**: Achieve resource efficiency
- **Rapid scale**: Scale easily to meet demand

Demo: Run first container

How to containers run

Containers: Introduction

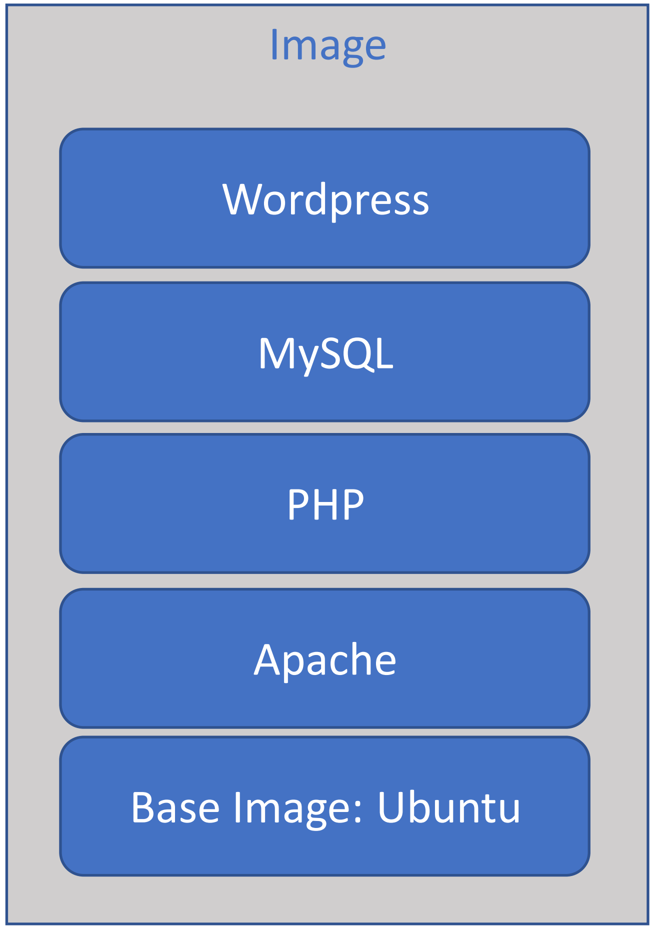


Images: Introduction

What is a container image?

Images: Introduction

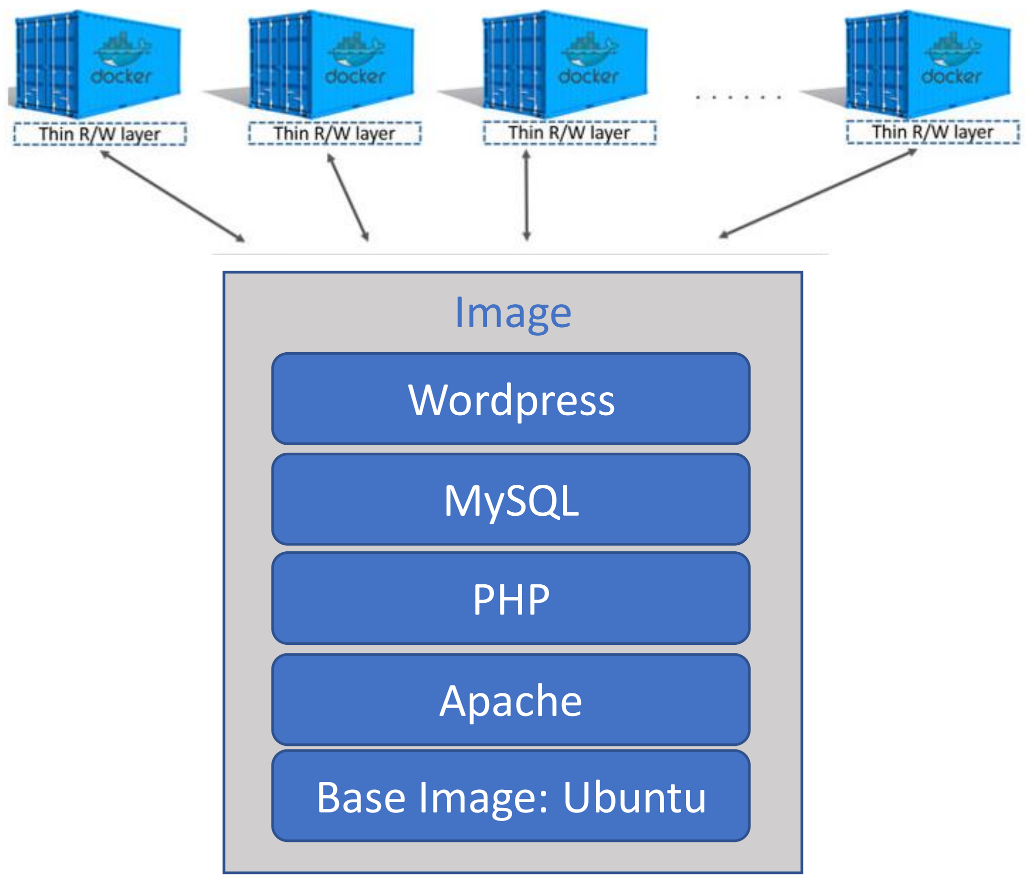
- Read-only templates for containers
- Can depend on other images
- Built up from a series of layers
- Initial layer is called base image
- Need to carefully choose base image
- For every change made on base image a new layer is created



How container runs?

Images: Introduction

- Each container has its own writable container layer
- All changes are stored in this container layer
- Multiple containers can share access to the same underlying image but have their own data state
- Image to be used needs to be on local cache



How Image and Container relates?

Images: Introduction

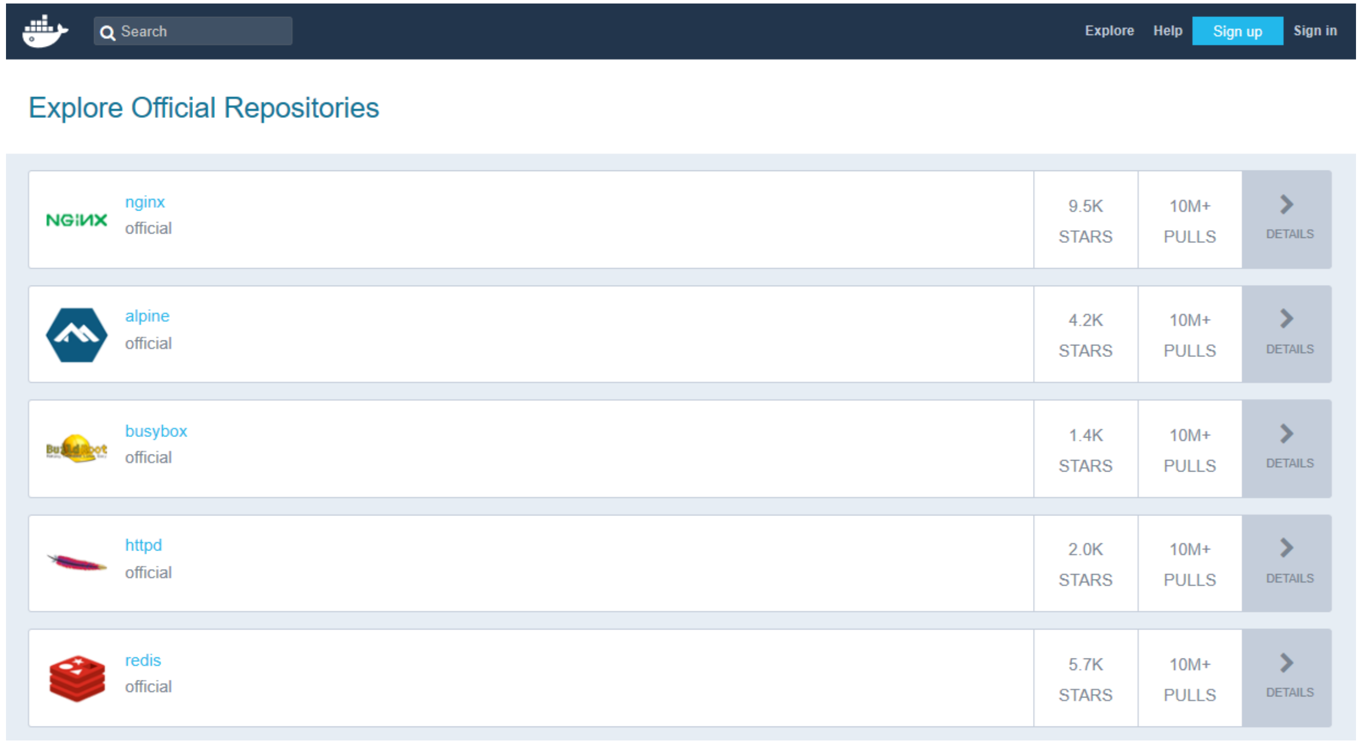
- Image is a template for the container
- Container is a running instance of the workload
- Making VMs comparison
  - Image is VHD + Config
  - Container is the running VM
- Making OOP comparison
  - Image is a class
  - Container is an instance of the class (i.e. an object)
- Using one Image you can instantiate several containers

Registry: Introduction

What is a Registry?

Registry: Introduction

- Registry is a stateless, highly scalable server side application that stores and lets you distribute images



Explore Official Repositories

nginx official	9.5K STARS	10M+ PULLS	DETAILS
alpine official	4.2K STARS	10M+ PULLS	DETAILS
busybox official	1.4K STARS	10M+ PULLS	DETAILS
httpd official	2.0K STARS	10M+ PULLS	DETAILS
redis official	5.7K STARS	10M+ PULLS	DETAILS

How to use a a Registry?

Registry: Introduction

- Tightly control where your images are being stored
- Fully own your images distribution pipeline
- Integrate image storage and distribution tightly into your in-house development workflow
- Public registry and/or Private registry

Public vs Private

Registry: Introduction

- Public Registry
  - Allow pull images publicly
  - For push images you need to have permission
  - Example: Docker Hub and Docker Store
- Private Registry
  - Pull and push tasks are made under permission set
  - Same API and Tools as Docker Hub/Store/Registry
  - Can be installed on-prem
  - Example: Azure Container Registry, GitHub Packages

Docker: Container Lifecycle

What is Docker?

Docker: Container Lifecycle

Open-source software to build and manage containers.

Docker separates the application from the infrastructure using container technology

“Dockerized” apps can run anywhere on anything

No more dependency daemons so developers and system admins unite

What is Docker?

Docker: Container Lifecycle

podman

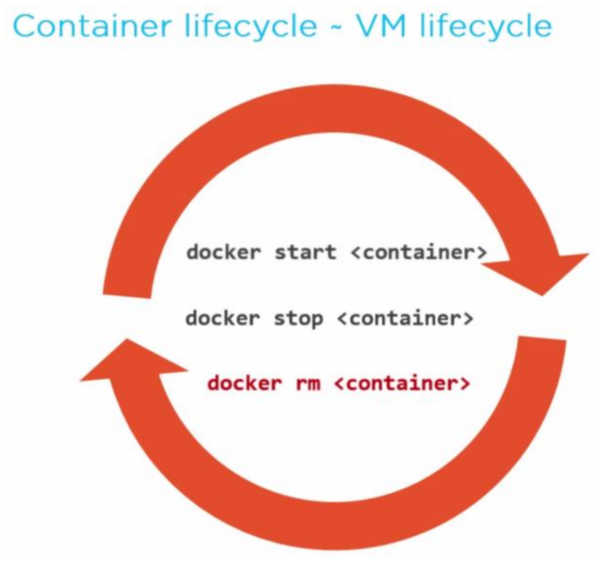
OPEN CONTAINER INITIATIVE

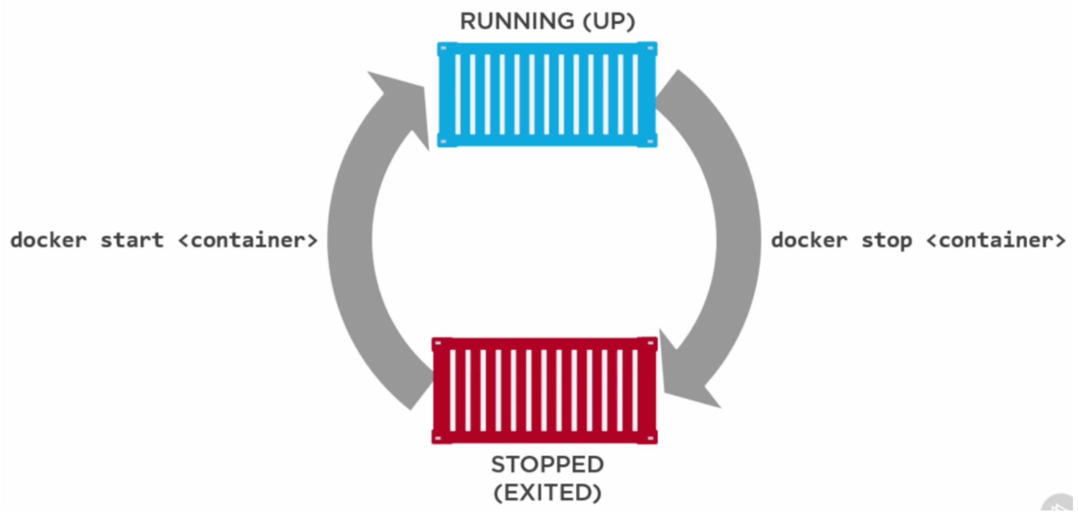
containerd

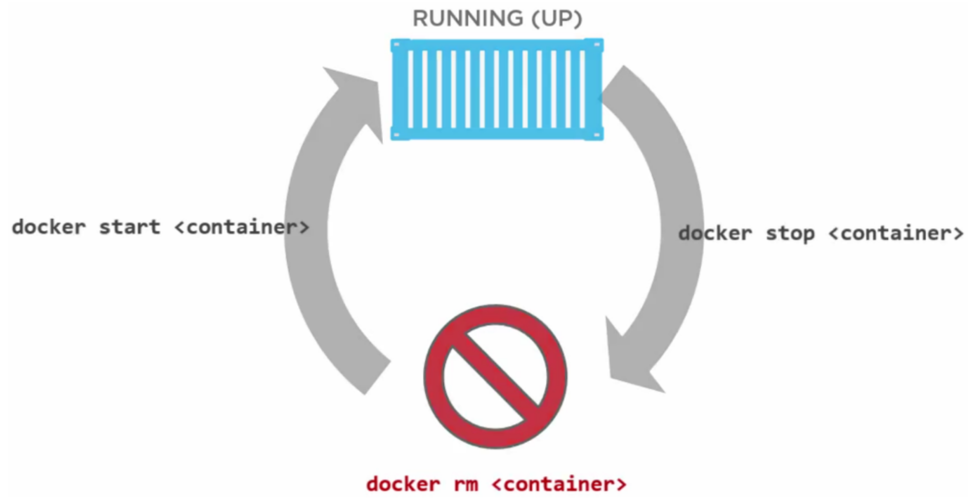
cri-o

Docker: Container Lifecycle

Docker: Container Lifecycle

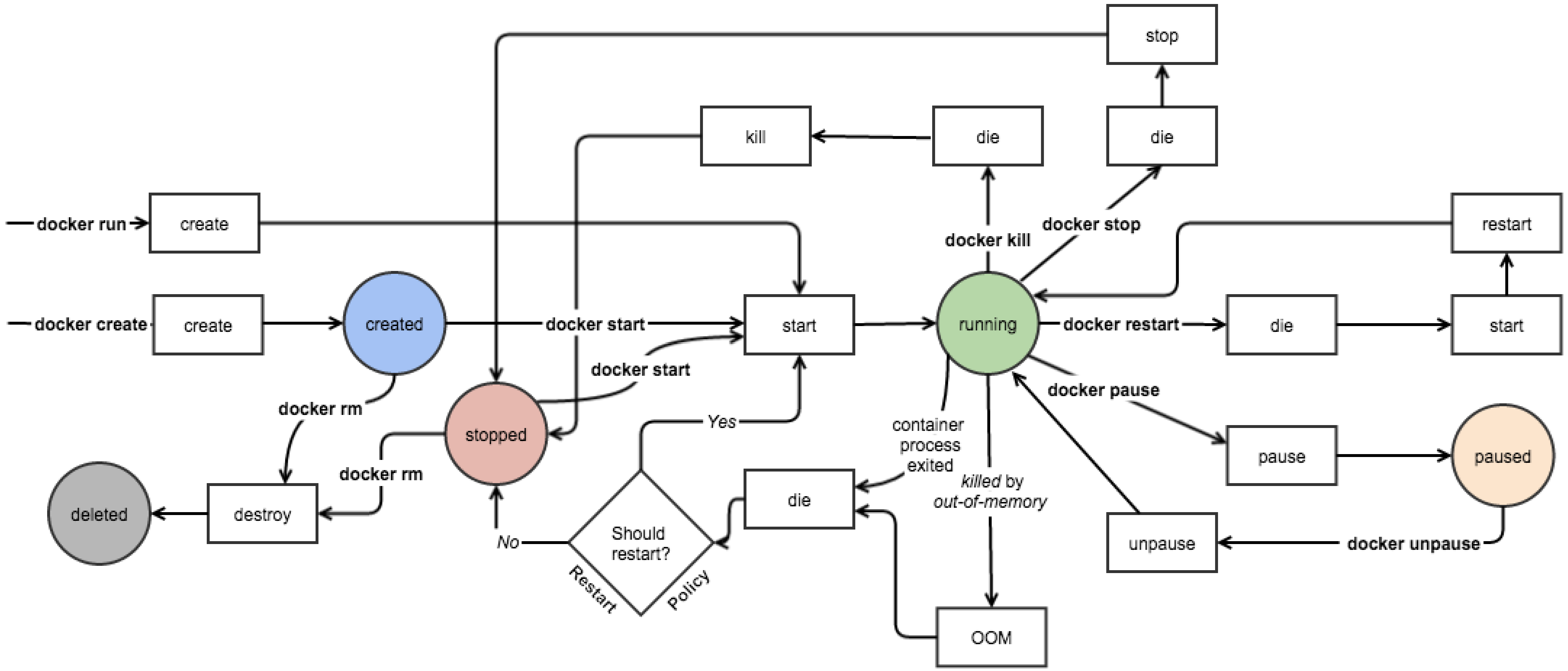






Docker: Container Lifecycle

Docker: Container Lifecycle



Demo: Container Lifecycle

Docker commands

Docker: Container Lifecycle

docker run -> Runs a command in new container

docker start -> Start one or more stopped containers

docker stop -> Stop one or more running containers

docker images -> List images

docker ps -> List Docker containers.

docker rm -> Remove one or more containers

docker rmi -> Remove one or more images

docker pull -> Pull an image or a repository from a registry

docker push -> Push an image or a repository to a registry

docker search -> Search the Docker Hub for images

Linux vs Windows Containers

Linux Containers

Linux vs. Windows Containers

- Containers started to be available only on Linux hosts with Linux Containers
- Now you may use Docker Desktop to manage and handle containers on Windows Host
- Windows Host can run Linux Containers using VMs or (better approach) WSL 2
- Windows Subsystem for Linux 2 allow you to run Linux inside Windows

Windows Containers

Linux vs. Windows Containers

- For running Windows Containers you need to have docker running on Windows Host
- Docker Desktop is a standard solution for developer machine (now with licensing…)
- For production environments you need to enable Containers feature on Windows Server (native on 2019 and 2022)
- Windows Container version needs to be equal or less than Windows Host Machine Kernel

Windows Containers

Linux vs. Windows Containers

Windows (https://hub.docker.com/_/microsoft-windows) *New in Windows Server 2019

- Automation workloads
- Carries most Windows OSS components

Windows Server Core (https://hub.docker.com/_/microsoft-windows-servercore)

- Minimal installation of Windows Server 2016
- Contains only core OS features
- Command-line access only

Nano Server (https://hub.docker.com/_/microsoft-windows-nanoserver)

- Available only as container base OS image (no VM support)
- 20 times smaller than Server Core
- Headless – no logon or GUI
- Optimized for .NET Core applications

Lab

Lab 1: Container Lifecycle

Github

Navigate to https://tasb.github.io/docker-kubernetes-training/

Read README.md for more details about the repo

Lab 01 - Introduction to containers | docker-kubernetes-training (tasb.github.io)