INTRODUCTION TO **Docker**

AGENDA

- Introduction
- What is a container?
- Why do containers exist?
- Containers vs. VMs
- System Containers vs. Application Containers
- Containers vs. Container Images
- Container technologies
- What is Docker?
- Why Docker?
- Docker engine components
- How does Docker work?
- Docker Buildfile
- Docker - Exercise

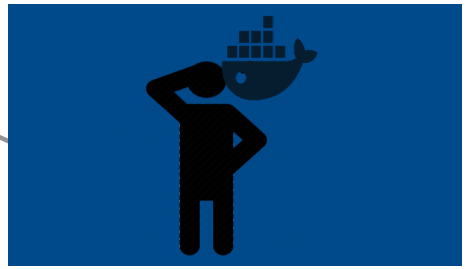
AGENDA

- Introduction
- What is a container?
- Why do containers exist?
- Containers vs. VMs
- System Containers vs. Application Containers
- Containers vs. Container Images
- Container technologies
- What is Docker?
- Why Docker?
- Docker engine components
- How does Docker work?
- Docker Buildfile
- Docker - Exercise

Introduction

Containers everywhere!

What are they and why do we need them?



How to build my own Docker container?

Why Docker? Are there other technologies at all?

AGENDA

- Introduction
- What is a container?
- Why do containers exist?
- Containers vs. VMs
- System Containers vs. Application Containers
- Containers vs. Container Images
- Container technologies
- What is Docker?
- Why Docker?
- Docker engine components
- How does Docker work?
- Docker Buildfile
- Docker - Exercise

What is a container?

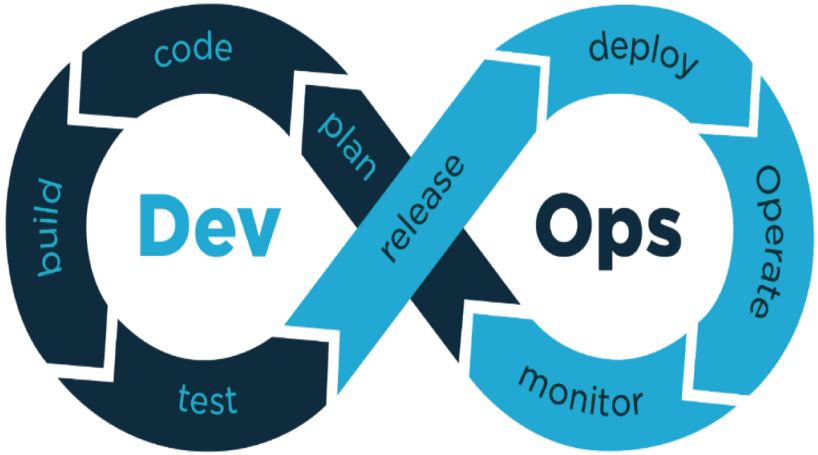
Isolated working environment for an application, containing all the necessary dependencies, libraries, binaries and configurations needed for the application to run seamlessly

AGENDA

- Introduction
- What is a container?
- Why do containers exist?
- Containers vs. VMs
- System Containers vs. Application Containers
- Containers vs. Container Images
- Container technologies
- What is Docker?
- Why Docker?
- Docker engine components
- How does Docker work?
- Docker Buildfile
- Docker - Exercise

Why Do Containers Exist?

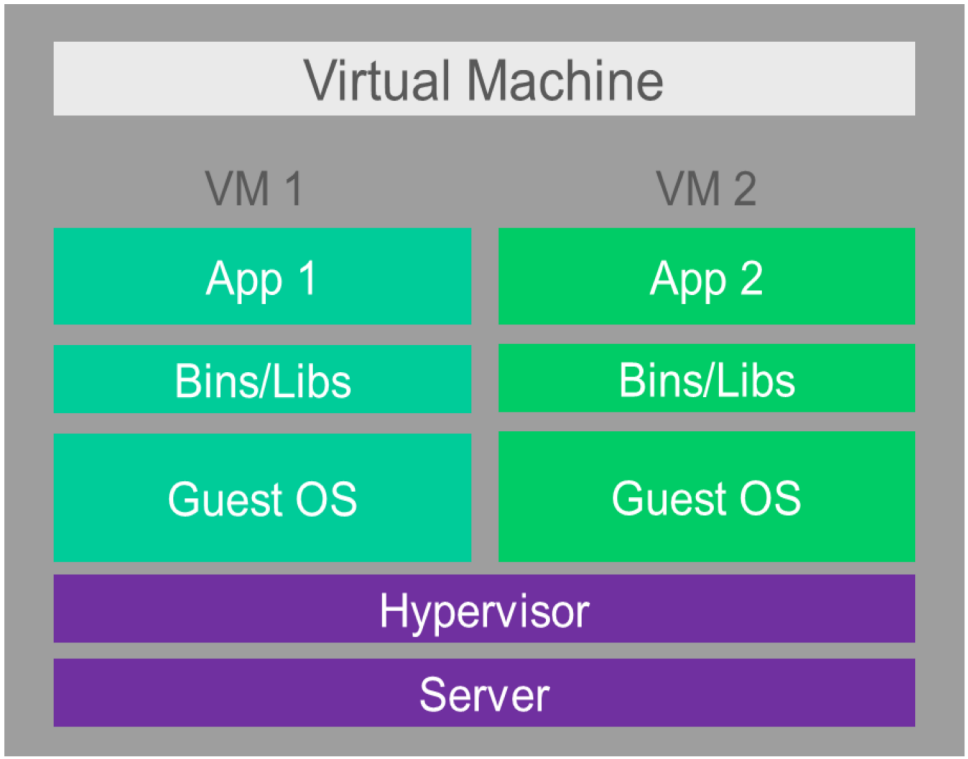
- Abstraction
- Isolation
- Portability
- Security
- Agility

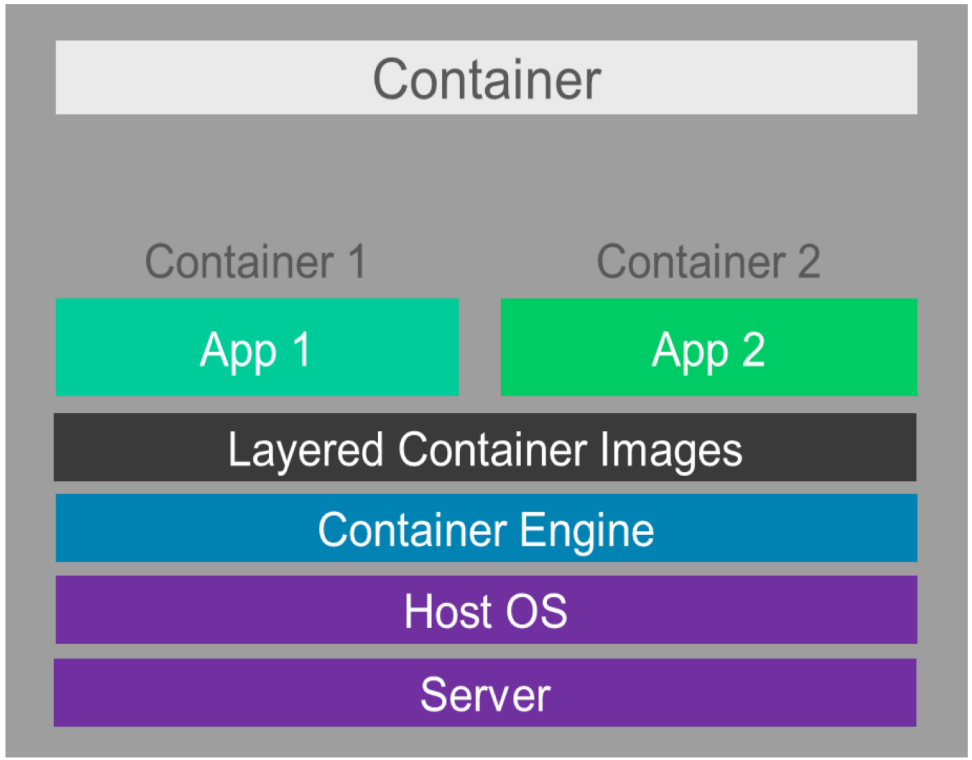


AGENDA

- Introduction
- What is a container?
- Why do containers exist?
- Containers vs. VMs
- System Containers vs. Application Containers
- Containers vs. Container Images
- Container technologies
- What is Docker?
- Why Docker?
- Docker engine components
- How does Docker work?
- Docker Buildfile
- Docker - Exercise

Container vs. Virtual Machine \1





Container vs. Virtual Machine \2

Parameter	Virtual Machines	Containers
Guest OS	Each VM runs on virtual hardware and Kernel is loaded into in its own memory region	All the guests share same OS and Kernel. Kernel image is loaded into the physical memory
Communication	Will be through Ethernet Devices	Standard IPC mechanisms like Signals, pipes, sockets etc.
Security	Depends on the implementation of Hypervisor	Mandatory access control can be leveraged
Performance	Virtual Machines suffer from a small overhead as the Machine instructions are translated from Guest to Host OS.	Containers provide near native performance as compared to the underlying Host OS.
Isolation	Sharing libraries, files etc between guests and between guests hosts not possible.	Subdirectories can be transparently mounted and can be shared.
Startup time	VMs take a few mins to boot up	Containers can be booted up in a few secs as compared to VMs.
Storage	VMs take much more storage as the whole OS kernel and its associated programs have to be installed and run	Containers take lower amount of storage as the base OS is shared

Container vs. Virtual Machine \3

Metaphor	Technical	Example
Harbour	(Virtual) machine	Laptop, AWS Instance
Cranes, Workers, Managers	Docker engine	CLI commands
Freight container	Docker container	Docker container
Container blueprint	Docker image	Dockerfile, Remote image
Shelf to keep the container blueprints	Docker registry	Docker Hub, private registry
Furniture	Depended Software and Container configuration	JDK8, NodeJS, PortMapping
Person who works in the container	Your software	JAR/WAR/EAR file

AGENDA

- Introduction
- What is a container?
- Why do containers exist?
- Containers vs. VMs
- System Containers vs. Application Containers
- Containers vs. Container Images
- Container technologies
- What is Docker?
- Why Docker?
- Docker engine components
- How does Docker work?
- Docker Buildfile
- Docker - Exercise

System Containers vs. Application Containers

Application Container

- Single entry point process
- Run a single application
- Built on top of system container technologies

System Container

- The entry point is an init system (Multiple services running on the same OS)
- Run different applications
- Built on native process resource isolation

AGENDA

- Introduction
- What is a container?
- Why do containers exist?
- Containers vs. VMs
- System Containers vs. Application Containers
- Containers vs. Container Images
- Container technologies
- What is Docker?
- Why Docker?
- Docker engine components
- How does Docker work?
- Docker Buildfile
- Docker - Exercise

Containers vs. Containers Images

Container

- Begin lifecycle using an image
- Running instance of an image
- Many containers can be run off the same image

Container Image

- Never started, never “running”
- Blueprint of a container (Inert file, that’s the base on which you instantiate containers)
- Ensure reusability of containers

AGENDA

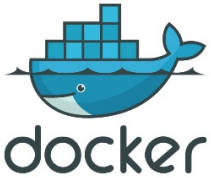
- Introduction
- What is a container?
- Why do containers exist?
- Containers vs. VMs
- System Containers vs. Application Containers
- Containers vs. Container Images
- Container technologies
- What is Docker?
- Why Docker?
- Docker engine components
- How does Docker work?
- Docker Buildfile
- Docker - Exercise

Container Technologies

Application Container

CF Warden/Garden

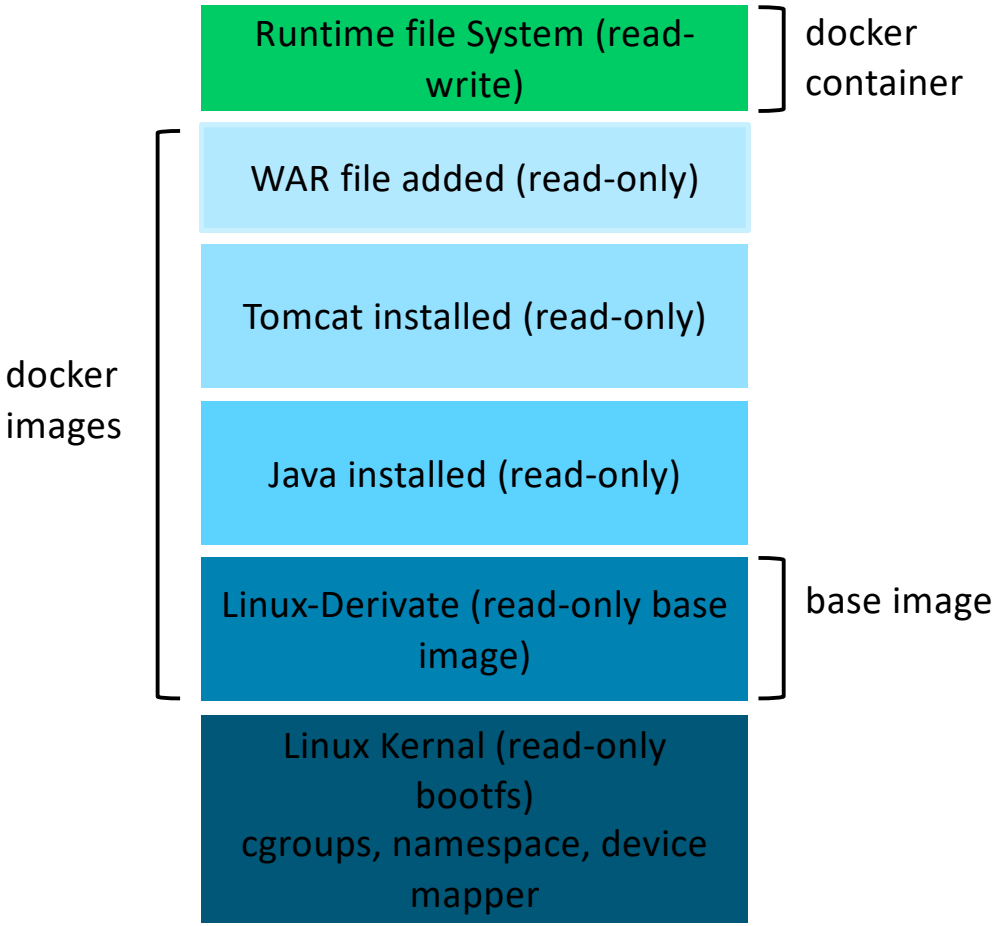
System Container



AGENDA

- Introduction
- What is a container?
- Why do containers exist?
- Containers vs. VMs
- System Containers vs. Application Containers
- Containers vs. Container Images
- Container technologies
- What is Docker?
- Why Docker?
- Docker engine components
- How does Docker work?
- Docker Buildfile
- Docker - Exercise

What is Docker?



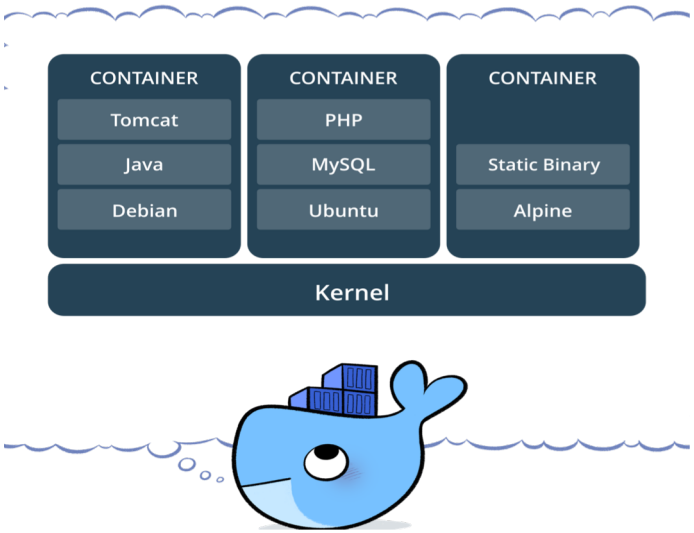
- A popular container with a broad support from the cloud community but also commercially:
  - **Linux-based docker** is based on cgroups and namespaces
  - **Windows-based docker** support has recently been integrated into Microsoft's Windows Server 2016. However, cgroups and namespaces are represented differently
- A single-(parent)-process container (unlike LXC) consisting of read-only layers. It's supposed to be stateless, such that when it is shut down, no data (state) must be lost

AGENDA

- Introduction
- What is a container?
- Why do containers exist?
- Containers vs. VMs
- System Containers vs. Application Containers
- Containers vs. Container Images
- Container technologies
- What is Docker?
- Why Docker?
- Docker engine components
- How does Docker work?
- Docker Buildfile
- Docker - Exercise

Why Docker?

- Lightweight nature
- Proper version control and component reuse
- Clear and clean documentation
- Easy to use
- Biggest and growing community
- Eliminates maintenance cost for enterprises
- Integrates with a number of orchestration tools
- Open source technology
- Homogeneity
- Scalability
- Interchangeability

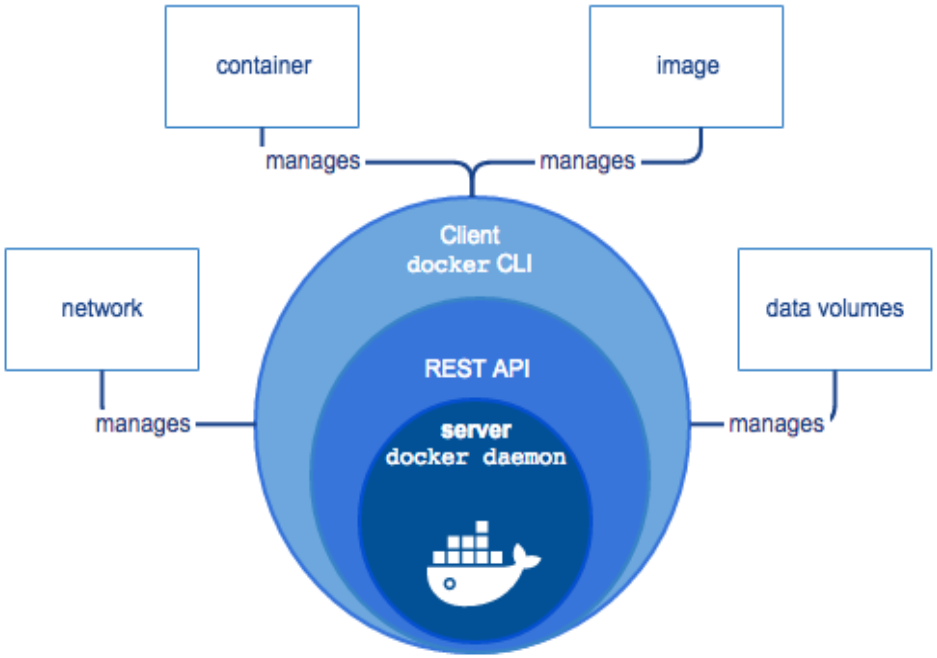


AGENDA

- Introduction
- What is a container?
- Why do containers exist?
- Containers vs. VMs
- System Containers vs. Application Containers
- Containers vs. Container Images
- Container technologies
- What is Docker?
- Why Docker?
- Docker engine components
- How does Docker work?
- Docker Buildfile
- Docker - Exercise

Docker Engine Components

- **Docker-Daemon:** A persistent background process (the dockerd command) that manages Docker images, containers, networks, and storage volumes. It constantly listens for Docker API requests and processes them (the dockerd command)
- **A REST API:** An API used by applications to interact with the Docker daemon. It can be accessed by an HTTP client
- **Docker-Client:** A command line interface (CLI) client (the docker command) for interacting with the Docker daemon



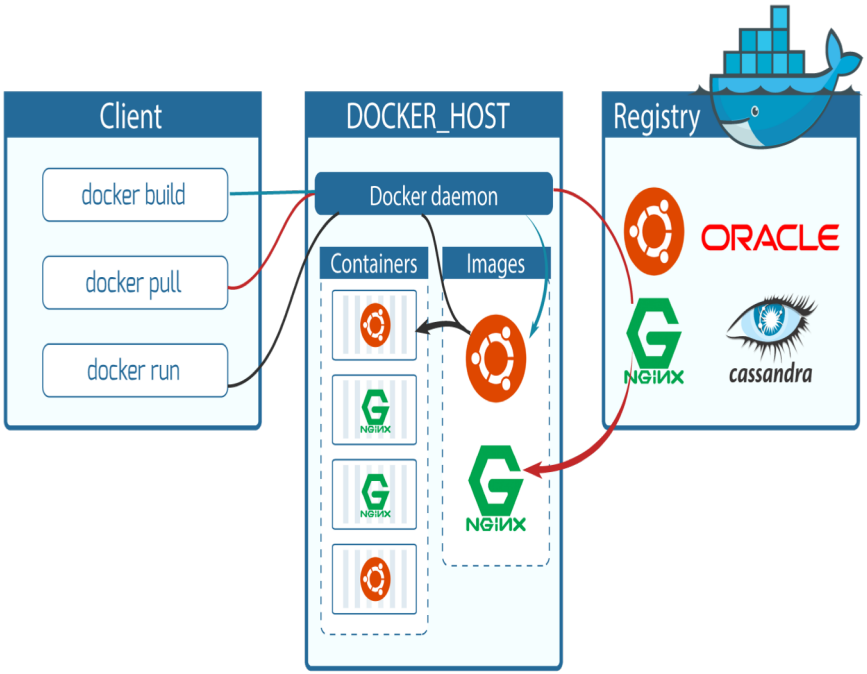
AGENDA

- Introduction
- What is a container?
- Why do containers exist?
- Containers vs. VMs
- System Containers vs. Application Containers
- Containers vs. Container Images
- Container technologies
- What is Docker?
- Why Docker?
- Docker engine components
- How does Docker work?
- Docker Buildfile
- Docker - Exercise

NOVATEC

How does Docker Work?

- Client requests for an Image
- Daemon checks if there is a local image
  - If there is no local image:
    - Get from registry (Docker Hub)
    - Download and store on PC
- Run and start container



AGENDA

- Introduction
- What is a container?
- Why do containers exist?
- Containers vs. VMs
- System Containers vs. Application Containers
- Containers vs. Container Images
- Container technologies
- What is Docker?
- Why Docker?
- Docker engine components
- How does Docker work?
- Docker Buildfile
- Docker - Exercise

Docker Buildfile

- Template for custom Container
- Based on a main Image (mostly a small Linux-Image)

- You can
  - Add Services
  - Define Variables
  - Execute commands
  - Execute file via
    - **docker build -t <filename>**

```
# Base image
from frolvlad/alpine-oraclejdk8:slim

# Adding a VOLUME at /tmp, because it's the Tomcat default working dir, created by Spring-Boot
volume /tmp

# add application artifact and rename it
add example.jar app.jar

# Make port 80 available to the world outside this container
EXPOSE 80

# Execute our application
RUN sh -c 'touch /app.jar'
ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/./urandom", "-Dspring.profiles.active=prod,h2", "-jar", "/app.jar"]
```

AGENDA

- Introduction
- What is a container?
- Why do containers exist?
- Containers vs. VMs
- System Containers vs. Application Containers
- Containers vs. Container Images
- Container technologies
- What is Docker?
- Why Docker?
- Docker engine components
- How does Docker work?
- Docker Buildfile
- Docker - Exercise

Docker - Exercise

- Complete part 1 and part 2 of
  - https://docs.docker.com/get-started/
- OR
  - https://docs.docker.com/docker-for-windows/
- In part 2: try to add a JAR file to the Buildfile instead of a Python

Questions?