Introduction to microservices with Docker

Sophie Chabridon

September 2024

Outline

1. Introducing Docker
2. Inside Docker
3. Containerized applications
4. Docker tools
5. Deployment challenges

1 Introducing Docker

- Deployment tool leveraging operating system container technology for building, shipping, and running programs
- Solves common software problems and simplifies installing, running, and managing software
- Docker runs everywhere
  - Available on Linux (with admin priviledges)
  - In a virtual machine for other OS like Windows or macOS
- Not a hardware virtualization technology. Docker interfaces directly with the host's Linux kernel
- The container technology allows to run programs in isolation
- Can be seen as a popular container technology for microservices

1.1 Docker and microservices

- A microservice is a application that implements a narrowly focused functionality
- A microservice is an autonomous deployable entity and can interact with other microservices
- It enables the continuous delivery and deployment of large, complex applications
- Package a microservice as a Docker container image and deploy each service instance as a container

1.2 Container engine of other technologies

- Docker is the container engine of several orchestrators
  - Kubernetes
    - Container orchestrator to run applications modeled as services across a cluster of hosts
  - Marathon from Apache Mesos
- Cloud solutions use Docker
  - Amazon EC2 Container Service
  - Google Container Engine
  - Microsoft Azure

1.3 A brief history of Docker

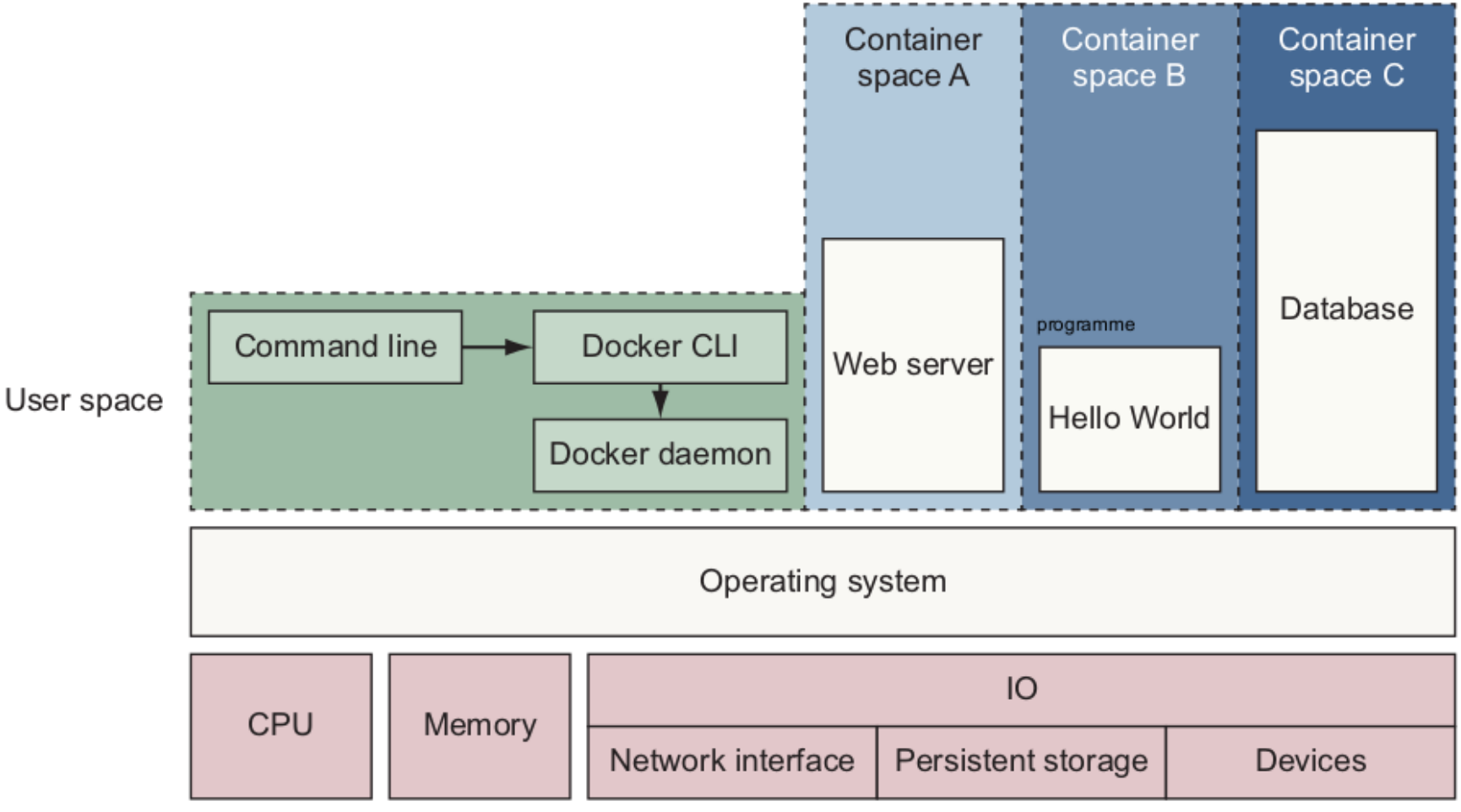
- 2013: Docker Inc. proposes their technology for packaging, shipping and running software as open-source
- Docker relies on the container mechanism (initially called jail) present in UNIX-style operating systems since the late 1970's
- Isolation: isolate a process from all resources except where explicitly allowed
- Containers are key for isolation but hard to build manually. Docker was proposed as a solution for DevOps
- Docker uses Linux namespaces and cgroups, which have been part of Linux since 2007

2 Inside Docker

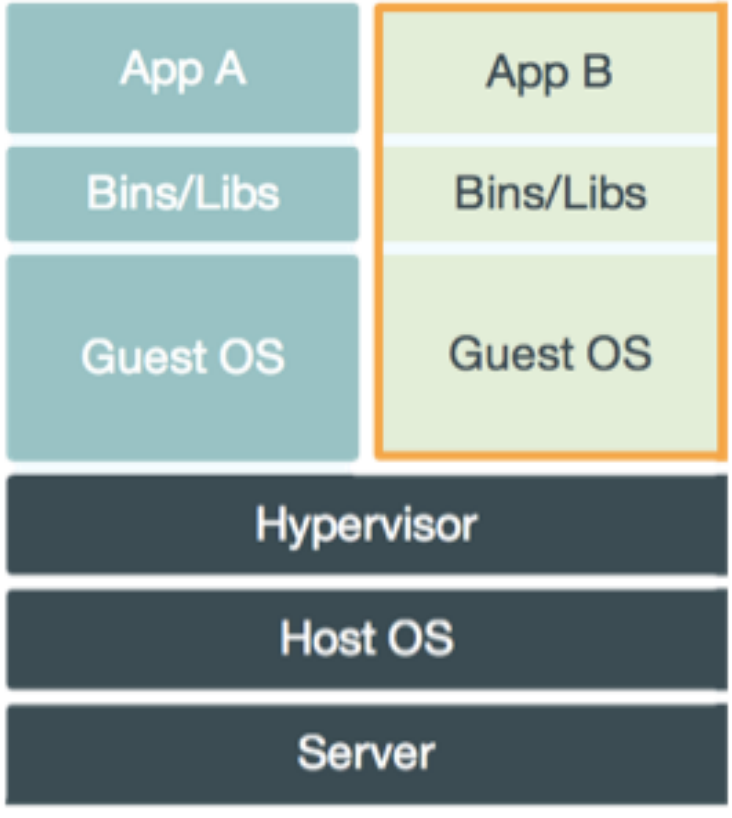
Docker makes use of 10 major system features

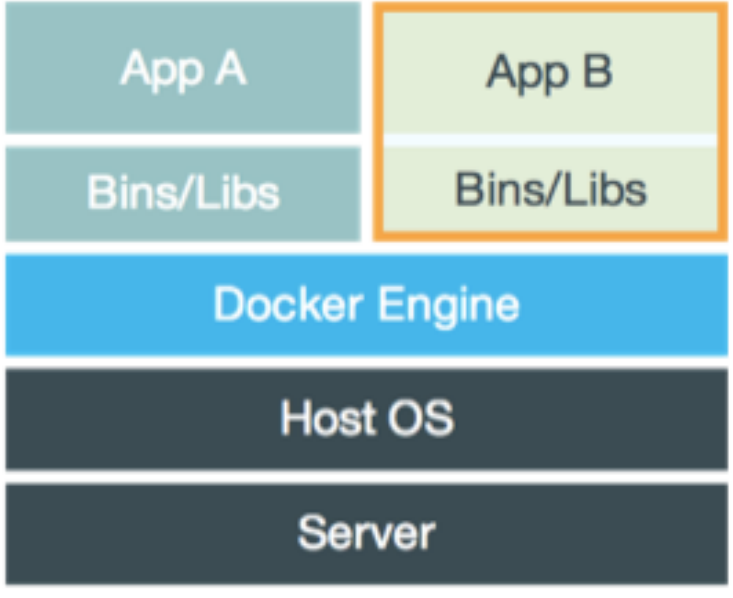
- PID namespace: Process identifiers and capabilities
- UTS namespace: Host and domain name
- MNT namespace: Filesystem access and structure
- IPC namespace: Process communication over shared memory
- NET namespace: Network access and structure
- USR namespace: User names and identifiers
- chroot syscall: Controls the location of the filesystem root
- cgroups: Resource protection
- CAP drop: Operating system feature restrictions
- Security modules: Mandatory access controls

3 Containerized applications



3.1 Virtual machine vs system container

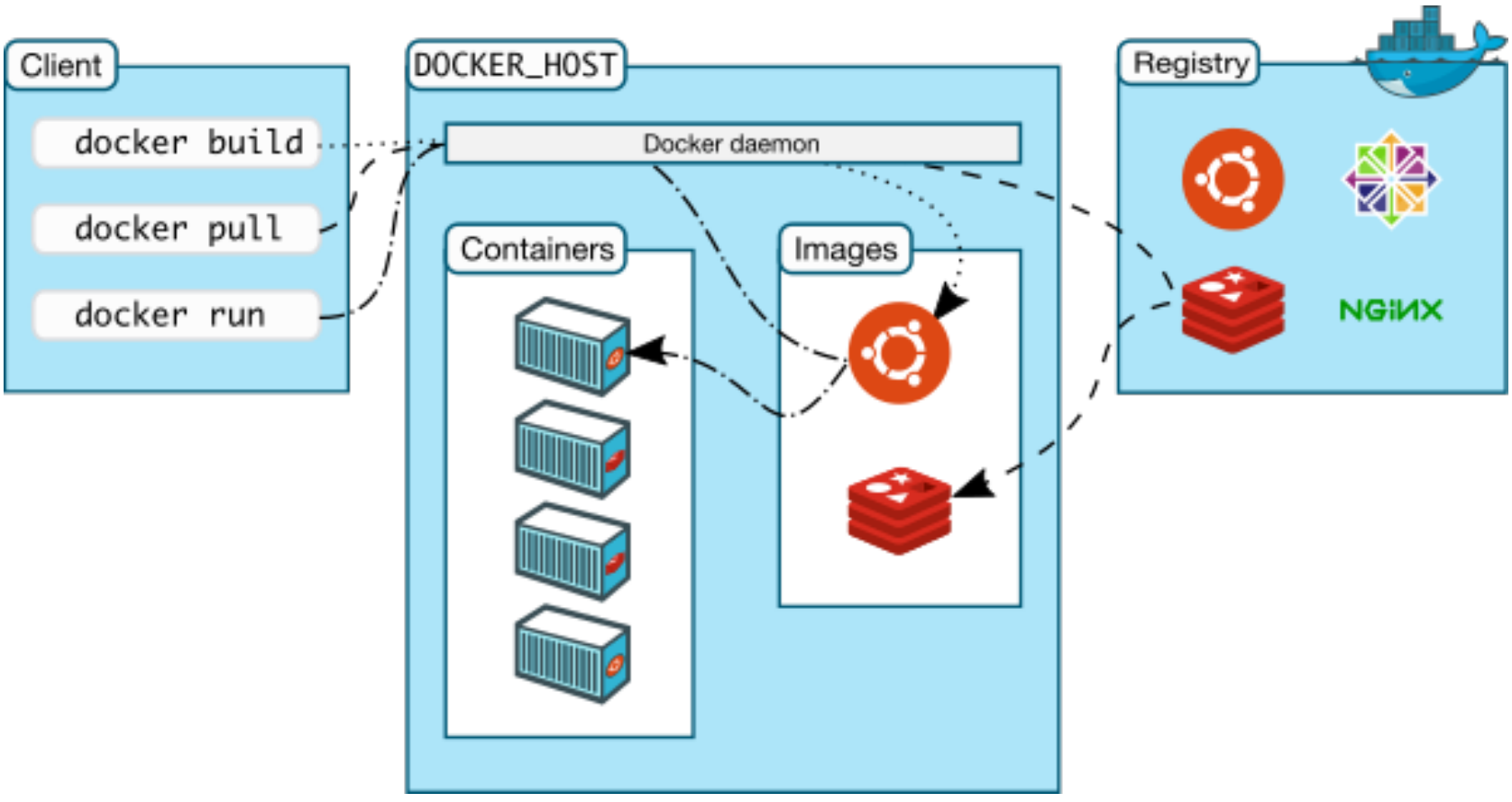




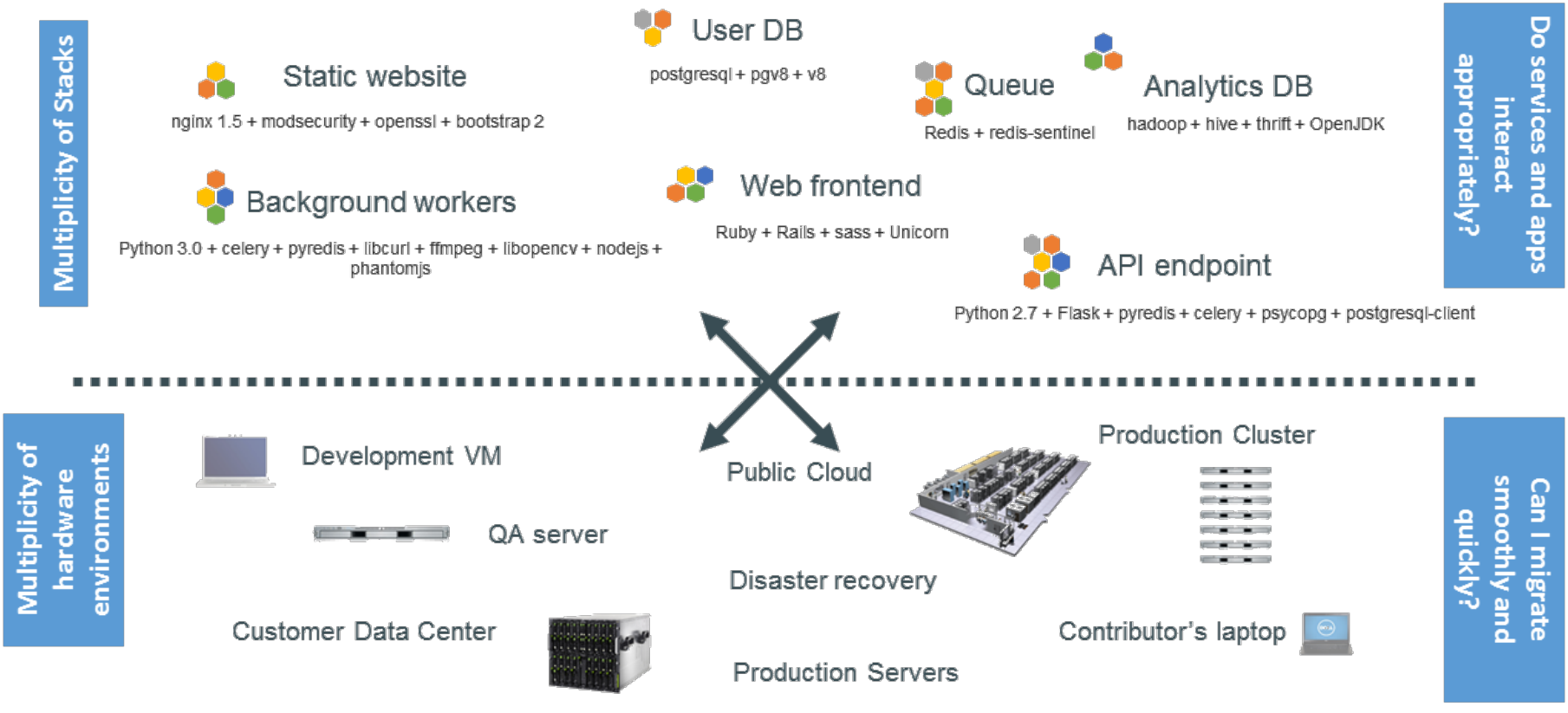
4 Docker tools

- Docker CLI: Commmand line user interface
- Docker Engine: Parent process of all docker containers created by the user
- Docker Compose: Declarative environment to build multi-service applications. Describe required state in a YAML file. Compose can model services, volumes, and networks.
- Docker Swarm: tool for clustering and orchestration. Adequate for small clusters, like in edge computing environments

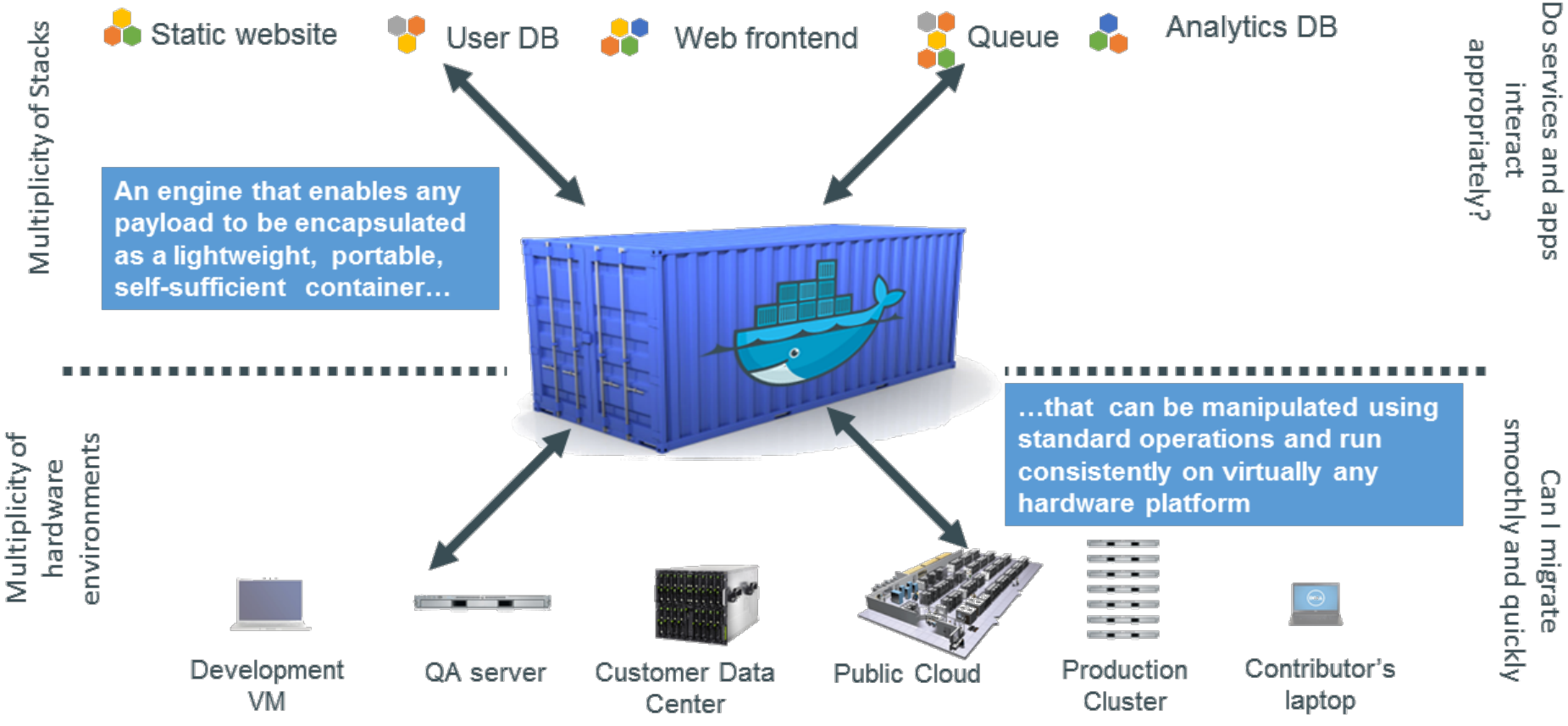
4.1 Docker client/server architecture



5 Deployment challenges



5.1 Deployment with Docker



References

Nickoloff, J. and Kuenzli, S. (2019).

Docker in Action, 2d edittion.

Manning Publications Co.

Richardson, C. (2019).

Microservices patterns.

Manning Publications Co.