

Use Cases of Container Technology

Peter von Rohr

1/18/2021

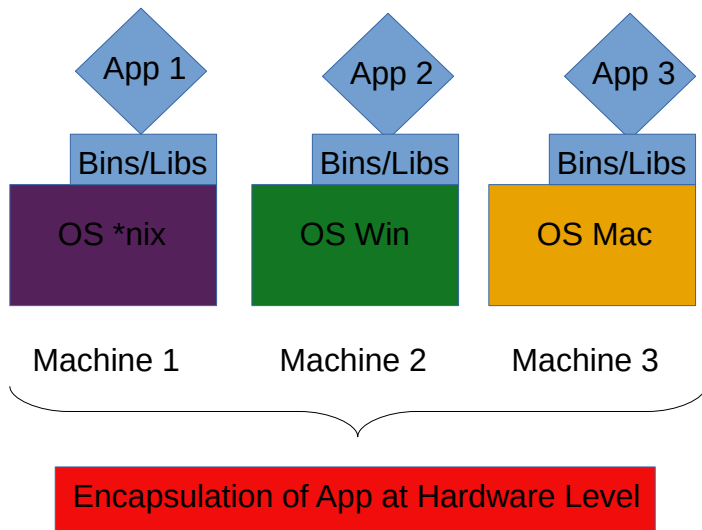
Background

- ▶ **Problem:** Multiple Applications (Apps) on the same machine can lead to
 - ▶ OS¹ Windows: DLL hell
 - ▶ OS *nix: library nightmare
- ▶ In general: dependencies between software components are difficult to manage
- ▶ Solution: encapsulation of software and libraries into independent units

¹Operating System

Solution 1: Hardware

One server/machine per App

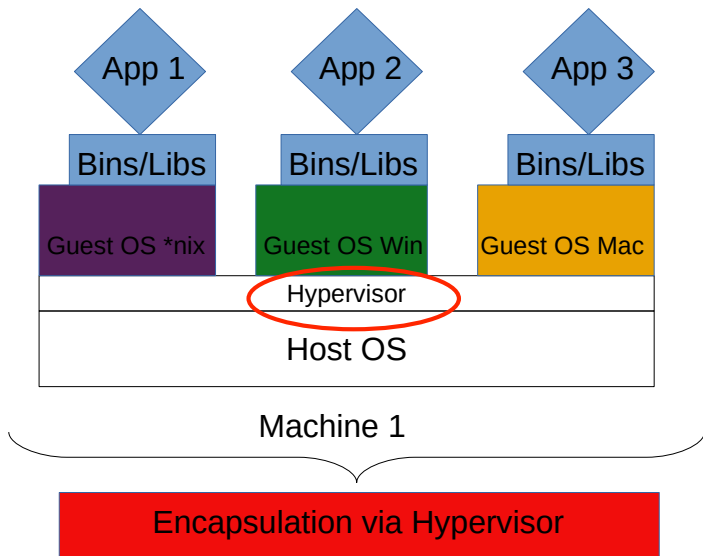


Problems with Solution I

- ▶ efficiency: hardware costs per app
- ▶ security: drag along old OS-versions (Win XP, Win 7, old *nix)
- ▶ flexibility: testing new apps, updating cycle of existing apps

Solution II: Virtualisation

Hypervisor Software as separation between host and guest

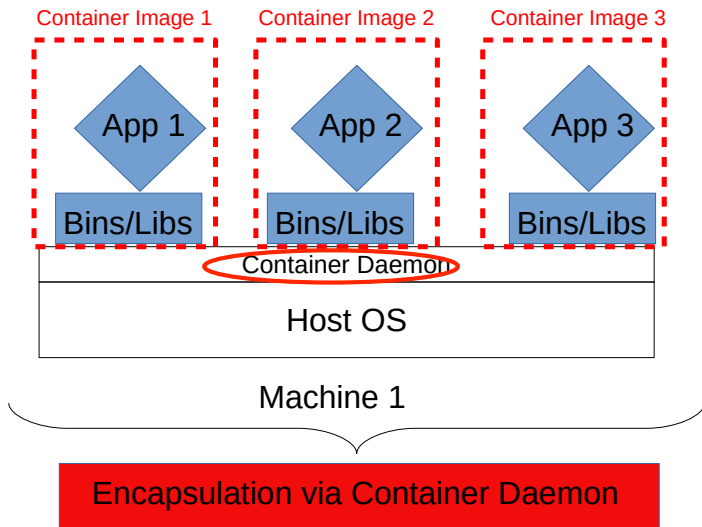


Problems with Solution II

- ▶ still just one app per (guest) OS
- ▶ intensive in resource requirements
- ▶ redundancy in configurations and settings (e.g. network)

Solution III: Containers

Container daemon replaces hypervisor



Container Facts

- ▶ Full benefit only with linux OS
- ▶ With OS Win and OS Mac: closer to virtualisation
- ▶ Still
 - ▶ develop it once, run it somewhere else
 - ▶ great for testing and updating systems

Container Products

- ▶ Docker

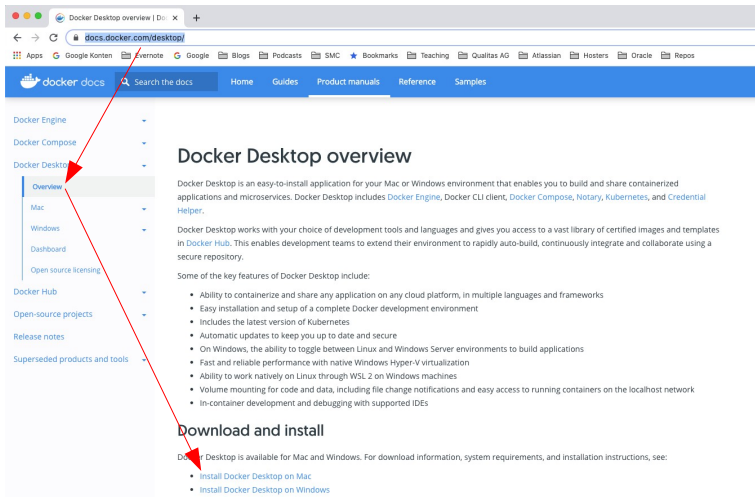
- ▶ best known
- ▶ industry standard
- ▶ deployment in the cloud
- ▶ requires root
- ▶ images in layers
- ▶ portability via dockerhub

- ▶ Singularity

- ▶ preferred in HPC environment
- ▶ can be run as non-root
- ▶ images in files
- ▶ portability via hubs (singularity and docker)
- ▶ can build images based on dockerfiles

Demo I: Getting Started with Docker

- ▶ On Mac/Win: Download docker desktop
(<https://docs.docker.com/desktop/>)



The screenshot shows the Docker Docs website with the URL docs.docker.com/desktop/ in the browser address bar. The left sidebar contains a navigation menu with the following items: Docker Engine, Docker Compose, Docker Desktop (expanded), Overview (selected), Mac, Windows, Dashboard, Open source licensing, Docker Hub, Open-source projects, Release notes, and Superseded products and tools. The main content area is titled "Docker Desktop overview" and contains the following text:

Docker Desktop is an easy-to-install application for your Mac or Windows environment that enables you to build and share containerized applications and microservices. Docker Desktop includes [Docker Engine](#), Docker CLI client, [Docker Compose](#), [Notary](#), [Kubernetes](#), and [Credential Helper](#).

Docker Desktop works with your choice of development tools and languages and gives you access to a vast library of certified images and templates in [Docker Hub](#). This enables development teams to extend their environment to rapidly auto-build, continuously integrate and collaborate using a secure repository.

Some of the key features of Docker Desktop include:

- Ability to containerize and share any application on any cloud platform, in multiple languages and frameworks
- Easy installation and setup of a complete Docker development environment
- Includes the latest version of Kubernetes
- Automatic updates to keep you up to date and secure
- On Windows, the ability to toggle between Linux and Windows Server environments to build applications
- Fast and reliable performance with native Windows Hyper-V virtualization
- Ability to work natively on Linux through WSL 2 on Windows machines
- Volume mounting for code and data, including file change notifications and easy access to running containers on the localhost network
- In-container development and debugging with supported IDEs

Download and install

Docker Desktop is available for Mac and Windows. For download information, system requirements, and installation instructions, see:

- [Install Docker Desktop on Mac](#)
- [Install Docker Desktop on Windows](#)

Docker Desktop Check

- ▶ Check whether docker desktop was installed

```
# show the version
```

```
docker version
```

- ▶ Run a first container image

```
docker run hello-world
```

Useful Docker Commands

```
# list of containers
docker ps -a
# remove container
docker rm <container_id>
# list of images
docker images
# remove images
docker rmi <image_name>
# download image from dockerhub
docker pull <image_name>
```

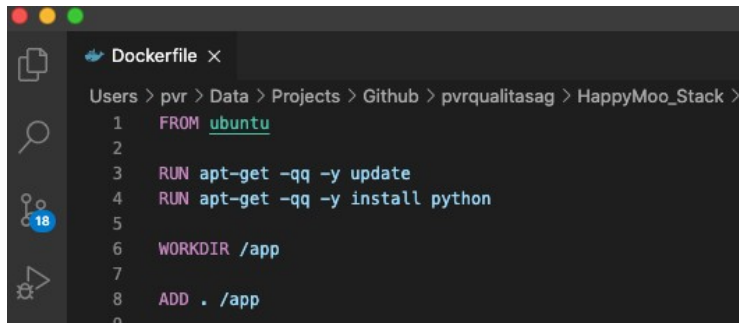
Dockerhub

- ▶ registry of public images: <https://hub.docker.com/>
- ▶ wide collection of docker images (e.g. ubuntu)
- ▶ try

```
docker pull ubuntu
```

Dockerfile

- ▶ Create own images via Dockerfile
- ▶ Example: run a python script

A screenshot of a code editor window titled "Dockerfile" with a close button. The editor shows a Dockerfile with the following content:

```
1 FROM ubuntu
2
3 RUN apt-get -qq -y update
4 RUN apt-get -qq -y install python
5
6 WORKDIR /app
7
8 ADD . /app
9
```

 The left sidebar of the editor shows a file explorer with a folder icon, a search icon, a git icon with a blue badge containing the number "18", and a play button icon. The breadcrumb path at the top of the editor reads: "Users > pvr > Data > Projects > Github > pvrqualitasag > HappyMoo_Stack >".

Build and Run

```
# build image from Dockerfile
docker build -t my_do_py .
...
# run
docker run -it my_do_py python /app/my_script.py
```

► Add the run command to the Dockerfile by

```
CMD ["python", "/app/my_script.py"]
```

Deployment

- ▶ Other server: 2-htz.quagzws.com
- ▶ Use the same Dockerfile for building
- ▶ Run with the same commands

Real World

- ▶ Rstudio form inside docker
- ▶ Dockerhub

```
docker run -d --rm -p 10087:8787 \  
  -e PASSWORD=yourpasswordhere rocker/rstudio
```



- ▶ Map home directory with option -v
 <host_dir>:<container_dir>

Singularity

- ▶ Reference: <https://sylabs.io/>
- ▶ Organises images in files
- ▶ Can be run as ordinary user
- ▶ Work with sandbox

Singularity Sandbox

```
sudo singularity build --sandbox ubuntu_s docker://ubuntu
```

- ▶ Write singularity file
- ▶ Run python program

```
singularity exec ubuntu_s/ python my_script.py
```

Update Sandbox

```
sudo singularity shell --writable ubuntu_s  
apt-get update -y  
apt-get install -y python
```

Singularity File

```
Bootstrap: docker
```

```
From: ubuntu
```

```
%post
```

```
    apt-get -y update
```

```
    apt-get install -y python
```

```
%files
```

```
    my_script.py /
```

```
%runscript
```

```
    python /my_script.py
```

► Build and run with

```
sudo singularity build ubuntu.sif Singularity.recipe  
./ubuntu.sif
```

Resources

- ▶ Container Explainer:
<https://www.youtube.com/watch?v=FWpnbGnzk08>
- ▶ Virtual Machines vs Docker Containers - Dive Into Docker:
https://www.youtube.com/watch?v=TvnZTi_gaNc
- ▶ Docker for beginners: <https://docker-curriculum.com/>
- ▶ Singularity: <https://sylabs.io/>
- ▶ Singularity Example Workflow:
<https://www.youtube.com/watch?v=m8lIDjFuXlc&t=15s>