HTWK Hochschule für Technik, Wirtschaft und Kultur Leipzig

IM Fakultät Informatik und Medien

Betriebliche Informationssysteme

Docker auf dem Raspberry Pi

28. JANUAR 2021 Sebastian Pötter, Mathis Greve, Fakultät Informatik und Medien

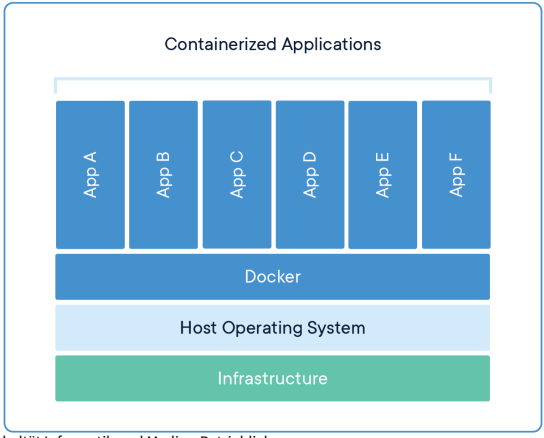
Gliederung

1. Docker
2. Raspberry Pi
3. .NET Core & C#
4. Fazit

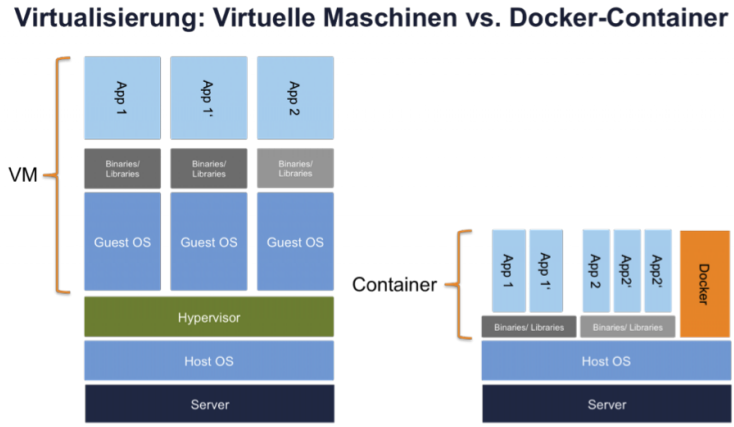
Docker - Allgemein

- Container für leichtgewichtige Virtualisierung
- Keine Emulation der Hardware, Isolierung mit namespaces
- Jeder Container eine abgekapselte Einheit
- Effizient Skalierbar
- Grundlage einer Applikation bildet eine Image-Datei

Docker - Aufbau

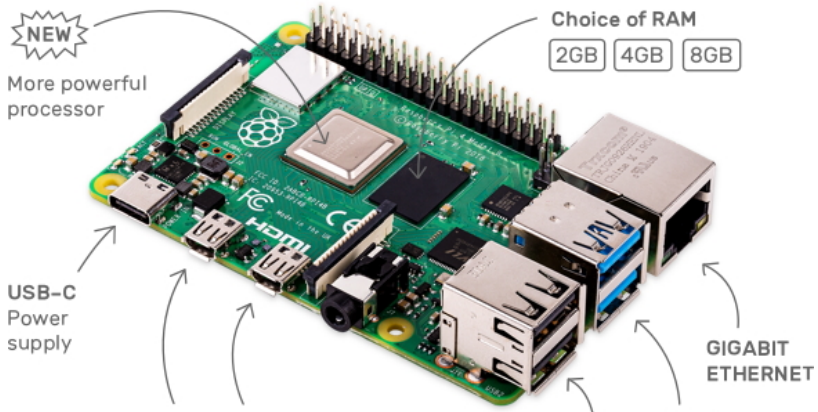


Docker und VM's



Quelle: Docker, Crisp Research, 2014

Raspberry Pi 4 - Aufbau



Raspberry Pi 4 - Spezifikationen

- 4x1.5GHz ARM CPU
- 2GB - 8GB LPDDR4-3200 SDRAM
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports
- 40 pin GPIO header
- 2 × micro-HDMI ports
- Micro-SD Karten slot
- 5V DC via GPIO header/USB-C connector, sowie PoE

Vorteile und Nachteile

Vorteile

- Kleiner und Handlicher ’Server’
- Speichererweiterung mit SSD oder HDD möglich
- Native Linux Unterstützung
- Einfache Anwendungseinrichtung
- Viele Schnittstellen
- verschiedene Betriebsysteme (Linux und Windows IoT)

Nachteile

- 4 CPU-Kerne mit begrenzter Leistung
- Maximal 8 gb Arbeitspeicher
- Speicher bis maximal 32 GByte (Micro SD)
- Benötigt ARM-Kompilierung jeder Anwendung

Docker und Raspberry Pi

- Speicherintensive Anwendungen vermeiden (Schreibgeschwindigkeit/Speichergröße)
- Durch ARM-Architektur nur entsprechende Images verwendbar
  - Begrenzte Auswahl, sowie höherer Aufwand
- Offizieller Raspbian Release ist 32-Bit Version
  - Beta Version der 64-Bit (Beta-Test) öffentlich
  - Verwendung von 64-Bit Anwendungen nur mit zusätzlichem Aufwand
- Möglichkeit Docker Schwarm mit mehreren Pi's zu realisieren

Raspberry Pi - Achtung mit dem Speicher

Durch speicherintensive Anwendung hervorgerufene Schäden

Sebastian Pötter, Mathis Greve,Fakultät Informatik und Medien,Betriebliche Informationssysteme

C# (.Net Core)

Architekturen

- (AMD) x32 Windows
- (AMD) x64 Windows, macOS, Linux
- ARM32 Windows, Linux
- ARM64 Linux

Betriebsysteme

- Linux: Ubuntu, Fedora, Debian, Alpine
- macOS: 10.13+
- Windows Client: 7, 8.1, 10

Beispiel: Dockerfile

Basis-image:
FROM mcr.microsoft.com/dotnet/core/runtime:3.1.9-buster-slim-arm32v7
Einstiegspunkt:
WORKDIR /app
Kopieren von Daten in das Docker-image:
COPY program/bin/ .
Befehle welche ausgeführt werden sollen
CMD ["./program"]
Für Debug Zwecke
ENTRYPOINT [/bin/bash]

Beispiel: Updater Skript

```
#!/bin/bash
cd /home/pi/Projects/docker/program/
git fetch
if git status | grep -q "behind"; then
        git pull
        dotnet build --configuration Release
        cd ..
        docker rm $(docker stop $(docker ps -a -q --filter ancestor=program:1.0 --format="{{.ID}}"))
        docker build --tag program:1.0 .
        docker container run program:1.0
fi
```

Zusammenfassung und Fazit

- Kleine Server-Anwendungen möglich
- Erweiterbarer Speicher
- Docker leicht installierbar
- .Net Core voll funktionsfähig kompilierbar
- Großes Angebot von ARM-Anwendungen (Kompilier-Möglichkeiten gegeben)
- Befehle und Beispielprojekt im Wiki-Eintrag

Bildquellen

- **Docker (F.4)** https://www.docker.com/sites/default/files/d8/2018-11/docker-containerized-appliction-blue-border_2.png
- **Docker (F.5)** https://www.docker.com/sites/default/files/d8/2018-11/docker-appliction.png
- **Pi (F.6)** https://www.raspberrypi.org/homepage-9df4b/static/raspberry-pi-4-labelled-2857741801afdf1cabeaa58325e07b58.png