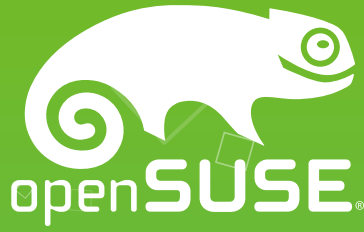


IBM zSystems Emulationen mit Container Anwendungen

Sarah Julia Kriesch

Accenture

sarah.julia.kriesch@accenture.com

AdaLovelace on LiberaChat

@sjkriesch

Resultate einer Bachelorarbeit im Berufsleben

Agenda

- Über mich
- Bachelorarbeit
- Situationen bei Kunden
- Wiederverwendung der Bachelorarbeit
- Rancher Desktop und Podman Desktop
- Emulationen von s390x in Kubernetes
- Docker BuildX versus Podman Buildah
- Integration in QEMU
- Demo
- Q & A

Über mich

- Sarah Julia Kriesch
- openSUSE Contributor seit 11 Jahren
- Member im openSUSE Release Engineering Team (s390x)
- Teamlead für s390x
- Gründer/Chair der Linux Distributions Working Group
- Bachelor Thesis bei IBM
- Consultant/Teamlead (+ Open Source Contributor) @ Accenture
- Studentin Master Informatik @ FAU
- IBM Champion 2023

Bachelorarbeit

- Enablement of Kubernetes Based Open-Source Projects for IBM Z
- Ziel: Bereitstellung von s390x Emulationen für Open Source Projekte
- Erstellung von Multi-Arch Container Images
- Verwendung von Docker BuildX
- Export vom Container Image und Start in QEMU

Kundensituationen

- Kein Zugriff auf das Mainframe für Externe
- Geschützte Umgebungen mit eingeschränkten (bis keinen) Rechten
- Tests in CI/CD Pipelines vor Übergabe an Kunden mit Mainframes

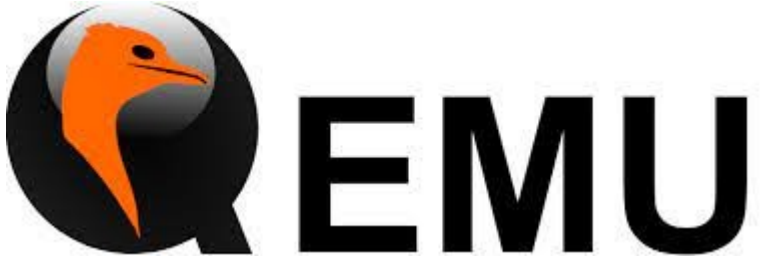
→ Emulationen auf dem Laptop und CI/CD

Wiederverwendung

- Emulationen in CI/CD Pipelines
- Integrierbar in Gitlab CI, Github Actions, Jenkins, ...
- Kubernetes Plattformen für Laptops (einfach installierbar):
  - Rancher Desktop
  - Podman Desktop

Emulationen

- Anwendungen für andere Architekturen auf der Host-Architektur laufen lassen (s390x auf x86)
- Unterschied zwischen User Mode Emulation und Full System Emulation



Full System Emulation

- Emulation des ganzen Systems (Hardware, Betriebssystem)
- Möglichkeit Priviledged Instructions zu verwenden
- Benötigt einiges an Hardware Ressourcen
- Nutzbar als Anwendungsentwicklungs-Plattform

User Mode Emulation

- Emulation von Anwendungsprozessen
- Verwendet nicht so viele Hardware Ressourcen
- Emulation von System Calls
- Nutzbar für einzelne Dienste

Nutzbar für Produktion?

- Emulationen benötigen zu viele Hardware Ressourcen
- Nur für Entwicklungs/Test-Zwecke geeignet
- Zu viele Prozesse/Instruktionen parallel
- Emulationen können die Auslastung mit Benutzern nicht wie ein richtiges Mainframe vertragen

Container Technologien

- Container Virtualisierung
- Anwendungen laufen isoliert innerhalb eines Containers
- Schnelles Deployment
- Jedes Jahr neue Entwicklungen an Container Engines

- Docker:
  - Container Engine
  - Grundlage der meisten Container Technologien
- Podman:
  - Container Engine
  - Nachfolger von Docker
- Kubernetes (K8S):
  - Container Orchestrierung
  - Skalierbarkeit

Rancher Desktop

- Kubernetes Plattform (Rancher) für den Laptop
- Installiert Kubernetes in VMs
- Support für Linux, Windows, MacOS
- Linux-Installation über OBS (deb/rpm/flatpak) und AppImage
- Link: https://rancherdesktop.io/

Podman Desktop

- Kubernetes Plattform (basierend auf Podman) für den Laptop
- Fast identische Features wie Rancher Desktop
- Support für Linux, Windows und MacOS
- Linux-Installation über Flatpak
- Link: https://podman-desktop.io/

Rancher Desktop versus Podman Desktop

- Unterstützung von Docker (RKE) und containerd (RKE2)
- Integriert K3S
- Verfügbar für alle Linux Distributionen
- Meistens einen Schritt voraus (war auch zuerst da)

- Basiert auf Features von Rancher Desktop
- Präferenz von Red Hat/Fedora
- Nur Podman als Container Engine verfügbar

Docker BuildX

- Enablement vom Bau von Multi-Arch Container Images
- Integrierte Emulation von alternativen Architekturen
- export DOCKER_BUILDKIT=1 (schon bei Installation aktiviert)

```
$ docker buildx create \
  --bootstrap \
  --name=kube \
  --driver=kubernetes \
  '--driver-opt="nodeselector=label1=value1,label2=value2","tolerations=key=key1,value=value1"'
```

```
docker buildx build \
--builder=kube \
--platform=linux/amd64,linux/arm64,linux/s390x \
-t skriesch/image-name \
--push
```

Doku: docs.docker.com/build/drivers/kubernetes

Pre-Requirement

- Installierte Pakete:
  - Docker
  - docker-buildx

Podman Buildah

- Basierend auf Docker BuildX
- Integrierte Emulation von alternativen Architekturen
- Bau von Container Images nur einzeln möglich

```
buildah manifest create multiarch
```

```
buildah bud \
--tag "hub.docker.com/skriesch/image-name:1.0" \
--manifest multiarch \
--arch amd64 \
.
```

```
buildah bud \
--tag hub.docker.com/skriesch/image-name:1.0 \
--manifest multiarch \
--arch s390x \
.
```

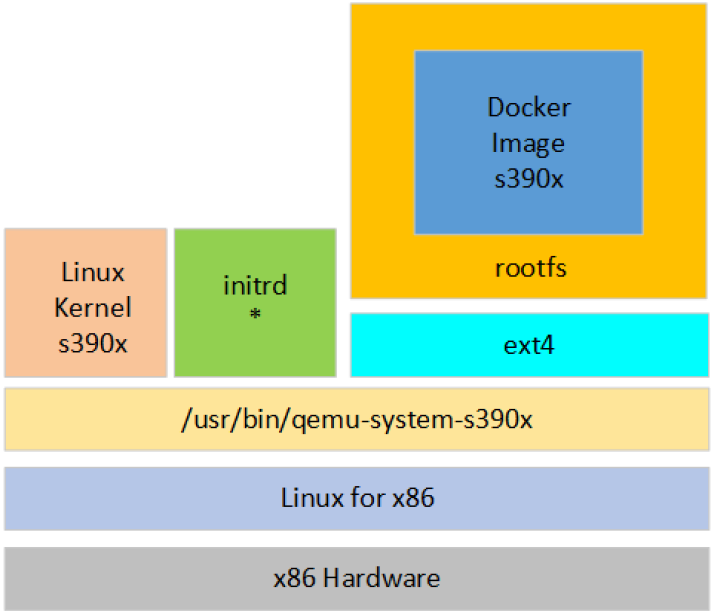
```
buildah manifest push --all \
multiarch \
"docker://hub.docker.com/skriesch/image-name:1.0"
```

Doku: https://danmanners.com/posts/2022-01-buildah-multi-arch/

Pre-Requirement

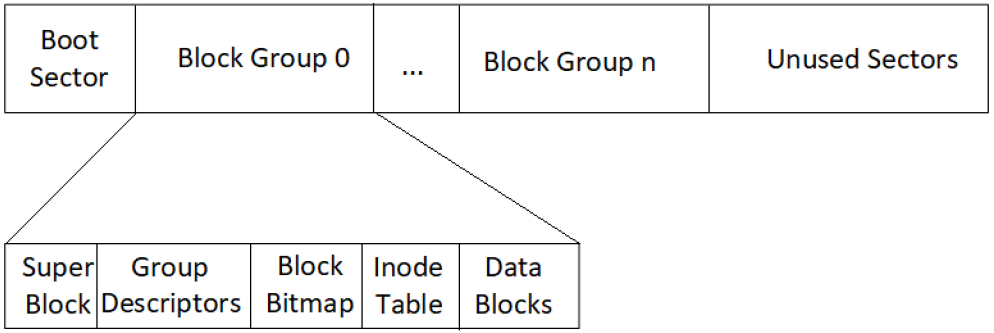
- Installierte Pakete:
  - Podman
  - buildah
  - qemu-extra u. qemu-linux-user (openSUSE)
  - qemu-user-static (RHEL/Fedora/Ubuntu)
  - qemu-arch-extra (ArchLinux)

Architektur mit QEMU

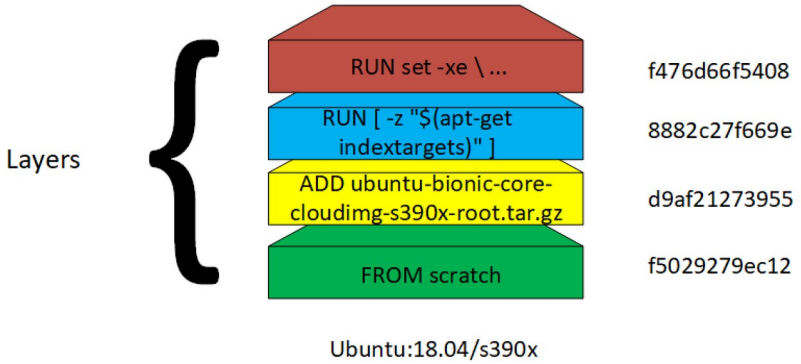


* optional

Dateisystem Kompatibilität



Ext4



UnionFS

- QEMU unterstützt nur Dateisysteme basierend auf Block Devices
- Umformatierung von UnionFS zu ext4 notwendig

Umformatierung des Dateisystems

- mkdir rootfs
- podman/docker export $(docker create image-s390x)|tar -C “rootfs” -xvf -
- qemu-img -f raw app.img $(docker images|grep ‘image-s390x’| ’{print int($7+0.5)"G"}’)
- mkfs.ext4 -F app.img
- mount -o loop app.img /mnt/rootfs
- cp -r rootfs/* /mnt/rootfs

System-Konfiguration

- Installation von qemu,qemu-linux-user, qemu-s390x, qemu-tools und docker
- Qemu bei DockerBuildX schon dabei
- Registrierung von s390x im Linux-Kernel-Modul binfmt_misc
- Möglich für mehrere Architekturen auf einem System

Binfmt_misc

- Mount binfmt_misc -t binfmt_misc /proc/sys/fs/binfmt_misc

```
echo ':qemu-s390x:M::\x7fELF\x02\x02\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x02\x00\x16:\xff\xff\xff\xff\xff\xff\xff\x00\xff\xff\xff\xff\xff\xff\xff\xff\
xfe\xff\xff:/usr/bin/qemu-s390x-static:OCF' > /proc/sys/fs/binfmt_misc/register
```

Registrierung mit Container Images

- Registrierung der Architektur durch laufende Container
- Docker und Podman nutzbar
- binfmt_misc integriert
- docker run --rm --privileged multiarch/qemu-user-static:register

- https://github.com/multiarch/qemu-user-static

Integration in QEMU

- Download eines Linux-Kernels oder Bau von einem
- Integration des konvertierten Dateisystems mit -hda
- Systemstart mit /bin/bash

```
/usr/bin/qemu-system-s390x -kernel bzImage -m 4G -M s390-ccw-virtio -nodefaults \
-device sclpconsole,chardev=console -parallel none -net none -chardev stdio,\
id=console,signal=off,mux=on -mon chardev=console -nographic -smp 3 \
-hda /data/cassandra.img  --append 'root=/dev/vda rw console=ttyS0 \
rdinit=/bin/bash'
```

Beispiel Github Actions

```yaml
name: QEMU to run s390x-focal

on:
  push:
  workflow_dispatch:

jobs:
  one:
    runs-on: ubuntu-latest
    steps:
    - name: Setup multiarch/qemu-user-static
      run: |
        docker run --rm --privileged multiarch/qemu-user-static:register --reset
    - name: ubuntu-core:s390x-focal
      uses: docker://multiarch/ubuntu-core:s390x-focal
      with:
        args: >
          bash -c
          "uname -a &&
          lscpu | grep Endian &&
          apt-get -y update &&
          apt-get -y install python3 git python3.8-venv &&
          python3 --version &&
          python3 -c 'import sys; print(sys.byteorder)' &&
          git clone https://github.com/simonw/sqlite-fts4 &&
          cd sqlite-fts4 &&
          python3 -m venv venv &&
          source venv/bin/activate &&
          pip install -e '.[test]' &&
          pytest
          "
```

Referenzen

- Bachelorarbeit:
  https://www.researchgate.net/publication/350958131_Enablement_of_Kubernetes_Based_Open-Source_Projects_on_IBM_Z

Questions?

Q & A


openSUSE®