Arbeit mit dem Docker Container für Sichere Programmierung

Prof. Dr. Christoph Karg*

26. Oktober 2025

1 Einleitung

Für die Vorlesung Sichere Programmierung ist eine virtuelle Maschine auf Basis der Intel 64-Bit Architektur erforderlich, in der die praktischen Teile der Lehrveranstaltung „ohne Risiko“ durchführbar sind. Da die Studierenden neben Intel-Rechnern vermehrt auch ARM-basierte Notebooks (z.B. Apple Silicon) nutzen, ist es wünschenswert, dass die virtuelle Maschine auf jeder dieser Architekturen lauffähig ist.

Der im weiteren Verlauf verfolgte Ansatz ist der Einsatz der Open Source Software QEmu[1]. QEmu (Quick Emulator) ist eine Software zur Emulation und Virtualisierung von verschiedenen Rechnerarchitekturen. Beispielsweise kann QEmu auf Apple Silicon (ARM-basiert) einen kompletten Rechner auf Intel 64-Bit Architektur zu emulieren. QEmu wird seit 2003 entwickelt und regelmäßig aktualisiert. Im Jahr 2025 erschien Version 10 der Software. QEmu ist eine etablierte Software und wird oft in Rechenzentren eingesetzt. Bei der Nutzung von QEmu im Lehrbetrieb gibt es folgende Herausforderungen zu lösen.

- Die Emulation ist im Vergleich zur Virtualisierung langsam. Emuliert man beispielsweise einen Intel-PC mit Linux und grafischer Oberfläche auf einem Apple Silicon Macbook Pro, dann ist damit kein vernünftiges Arbeiten möglich.
- Die Bedienung von QEmu ist nicht einfach. In der Regel nutzt man Befehle an der Kommandozeile, um die Emulation vorzubereiten und durchzuführen. Hinzu kommt, dass es Unterschiede in der Bedienung je nach dem eingesetzten Host-Betriebssystem gibt. Diese Umstände haben das Potential, die Studierenden zu überfordern.

Um diesen Punkten zu begegnen, wird die virtuelle Maschine über eine in einen Docker-Container eingebettete QEmu-Umgebung bereit gestellt. Die virtuelle Arbeitsumgebung

*Hochschule Aalen, Studiengang Informatik, Schwerpunkt IT-Sicherheit, E-Mail: christoph.karg@hs-aalen.de

[1] Webpage: https://www.qemu.org

besteht aus einer Debian-Installation ohne grafische Oberfläche. Der Zugriff auf die virtuelle Maschine erfolgt über die Secure Shell (SSH). Es werden Shell-Skripte zur Erstellung und zum Betrieb des Containers bereit gestellt. Dieses How-To erklärt die Installation und Nutzung dieses Containers. Voraussetzung für die Durchführung dieses How-Tos ist eine funktionierende unixoide Arbeitsumgebung (Linux, MacOS, Windows Subsystem for Linux) und eine Docker-Installation.

2 Erstellen des Docker-Containers

Die für die Erstellung des Containers benötigten Daten stehen unter folgendem Download bereit:

Webpage:	https://its.informatik.htw-aalen.de/virtual-machines/sipro
Nutzer:	`sipro`
Passwort:	`cheixi4ieVoo+r`

Es werden zwei Dateien bereit gestellt:

- `sipro-docker.tgz` ⇝ TAR-GZ-Archiv
- `sipro-docker.tgz.sha256sum` ⇝ Prüfsumme zur Überprüfung der Datenintegrität des Archivs

Zuerst werden die beiden Dateien herunter geladen. Um die Integrität des Archivs zu prüfen, nutzt man diesen Befehl:

```
> sha256sum -c sipro-docker.tgz.sha256sum
```

Ist die Prüfung nicht OK, dann muss das Archiv nochmals herunter geladen werden. Dies ist jedoch sehr unwahrscheinlich.

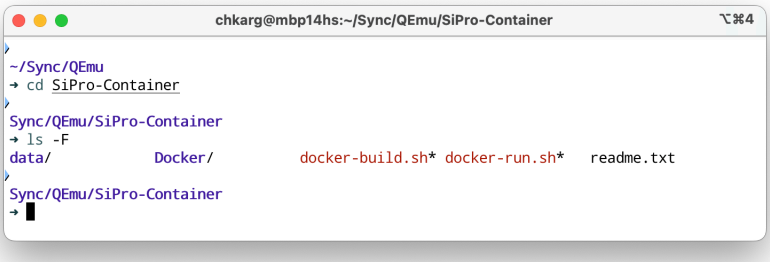
Mit diesem Befehl wird das Archiv entpackt:

```
> tar xvzf sipro-docker.tgz
```

Nach dem Entpacken des Archivs befinden sich die Daten in dem Verzeichnis `SiPro-Container`. Mit dieser Befehlsfolge wechselt man in das Verzeichnis und zeigt dessen Inhalt an:

```
> cd SiPro-Container
> ls -F
```

Das Ergebnis ist:



Das Verzeichnis enthält die folgenden Unterverzeichnisse:

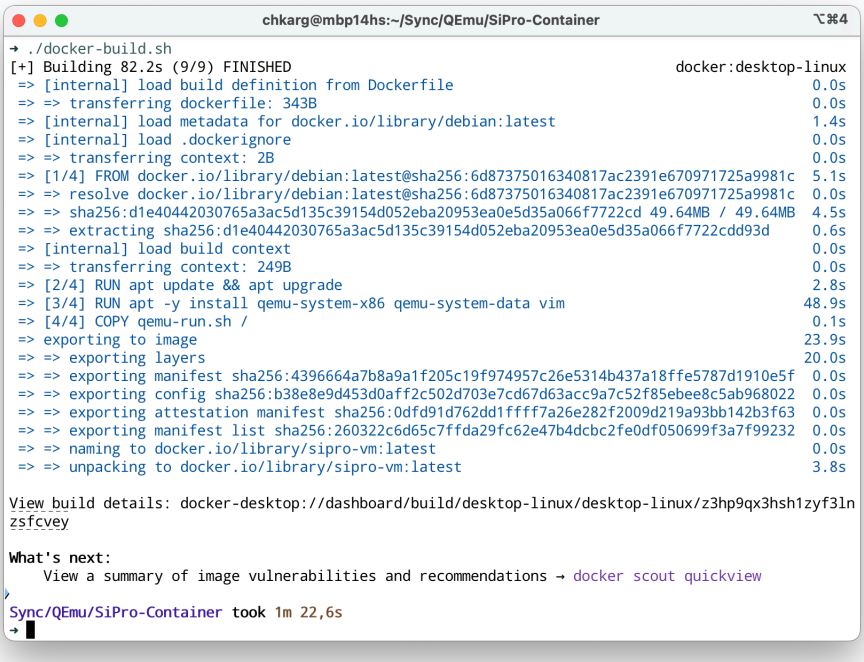
- `data` ⇝ enthält die virtuelle Festplatte der VM
- `Docker` ⇝ enthält die Daten zur Erstellung des Docker-Containers

Die beiden Shell-Skripte `docker-build.sh` und `docker-run.sh` dienen zur Erstellung beziehungsweise zum Ausführen des Docker-Containers.

Um das Image für den Docker-Container zu bauen, wird dieser Befehl ausgeführt:

```
> ./docker-build.sh
```

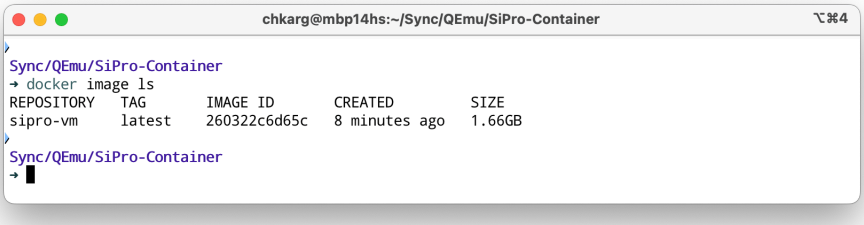
Je nach Leistungsfähigkeit des Computers dauert dies zwei bis fünf Minuten. Die Ausgabe im Terminal sieht in etwa so aus:



Anschließend zeigt der folgende Befehl Informationen zu dem erstellten Image an:

```
> docker image ls
```

Der Befehl liefert das folgende Ergebnis:



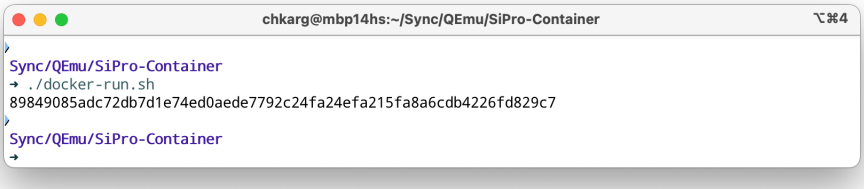
Der Name des Images ist demnach `sipro-vm`. Die Image-ID kann sich von der in obiger Abbildung unterscheiden.

3 Starten des Docker-Containers

Nach der Erstellung des Images kann der Container gestartet werden. Hierzu führt man folgenden Befehl in der Shell aus:

```
> ./docker-run.sh
```

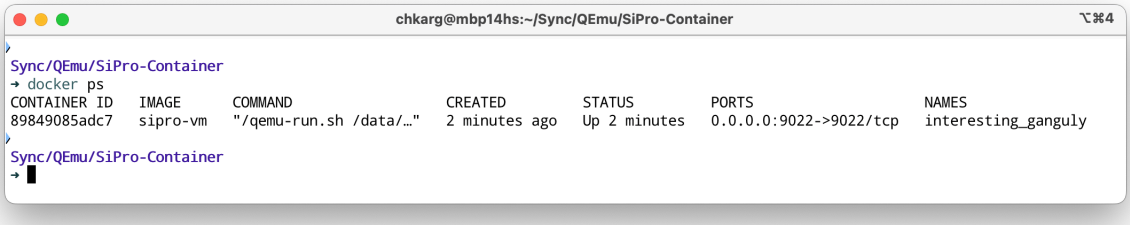
Die Ausgabe ist ähnlich zu dieser:



Das Docker-System startet den Container im Hintergrund. Die ausgegebene Nummer ist die ID des Containers. Um weitere Informationen anzuzeigen, gibt man folgenden Befehl ein:

```
> docker ps
```

Die Ausgabe ist ähnlich zu dieser:



Aus der Spalte PORTS ist ersichtlich, dass der Container einen offenen Netzwerk-Port 9022 besitzt.

Nach dem Start dauert es mehrere Minuten, bis der Container betriebsbereit ist. Im Innern des Containers wird via QEmu die virtuelle Maschine gebootet. An dieser Stelle wird empfohlen, eine Kaffee-Pause einzulegen.

4 Zugriff auf den Docker Container

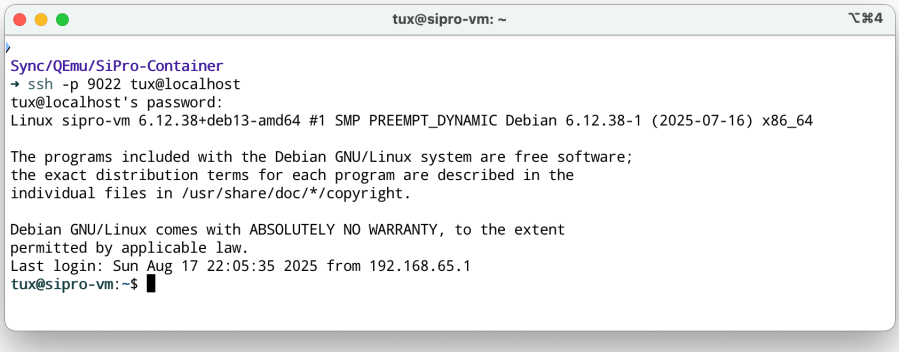
Die Zugangsdaten für die virtuelle Maschine sind:

Nutzer:	`tux`
Passwort:	`tux`

Um sich an der virtuellen Maschine anzumelden, wird die Secure Shell benutzt. Der Befehl zur Anmeldung lautet:

```
> ssh -p 9022 tux@localhost
```

Nach erfolgreicher Anmeldung befindet man sich in einer Login-Shell des Debian Trixie Betriebssystems.



Anschließend kann man mit der Arbeit am praktischen Teil der Vorlesung beginnen.

Um die virtuelle Maschine zu beenden, muss sie ordnungsgemäß herunter gefahren werden. Dies wird mit folgendem Befehl erledigt:

```
> sudo shutdown now
```

Nach dem Herunterfahren der virtuellen Maschine wird die Ausführung des Containers automatisch beendet.

5 SSH Konfiguration (optional)

In diesem Abschnitt wird gezeigt, wie man die SSH-Konfiguration so anpassen kann, um den Login-Vorgang an der virtuellen Maschine zu vereinfachen. Dieser Abschnitt ist optional.

Falls noch nicht geschehen, muss zunächst ein SSH-Schlüssel generiert werden. Der entsprechende Befehl lautet:

```
> ssh-keygen
```

Der SSH-Schlüssel enthält ein Paar bestehend aus einem öffentlichen und privaten Schlüssel. Der öffentliche Schlüssel wird nun in die virtuelle Maschine kopiert. Befehl:

```
> ssh-copy-id -p 9022 tux@localhost
```

Anschließend wird der private Schlüssel an den SSH-Agent übergeben. Befehl:

```
> ssh-add-key
```

Nun kann man sich an der virtuellen Maschine anmelden, ohne ein Passwort einzugeben:

```
> ssh -p 9022 tux@localhost
```

Die Anmeldung kann noch weiter vereinfacht werden, indem man die SSH-Konfiguration um folgenden Abschnitt erweitert:

```
Host siprovm
    Hostname localhost
    Port 9022
    User tux
```

Hierzu muss die Datei `config` editiert werden. Diese Datei befindet sich im Verzeichnis `$HOME/.ssh`. `$HOME` steht dabei für das Homeverzeichnis des angemeldeten Benutzers. Anschließend kann man sich mit folgendem Befehl an der virtuellen Maschine anmelden:

```
> ssh siprovm
```