

Dokumentation

Docker Compose auf Raspberry Pi

Andrea Furrer, Patrik Burkhalter

L-Tin-21-Fr-a

Netzwerktechnik, Thomas Engweiler

TEKO, März 2024

Version 1.0

Inhalte

Einleitung	2
Installation von Docker auf dem Raspberry Pi.....	3
Installation und Updates	3
Einrichtung von Docker Compose	4
Docker Compose Konfiguration	4
Docker Compose Konfiguration.....	5
Dienste	5
Netzwerke	6
Volumes.....	6
Zusammenfassung.....	6
Nutzung der Dienste.....	7
Samba-Server	7
Tomcat-Webserver	7
Jupyter Notebook-Server	7
Code	7

Einleitung

Diese Dokumentation gibt einen Einblick in die Einrichtung und Nutzung von Docker (Compose) auf einem Raspberry Pi für die Implementierung eines Samba-Dateiservers, eines Tomcat-Webserver und eines Jupyter Notebooks. Das Projekt wurde im Rahmen des Fachs "Netzwerk- und Betriebssysteme" entwickelt, um praxisnahe Erfahrungen im Umgang mit Containeranwendungen auf einem Raspberry Pi zu vermitteln.

Wir haben uns für folgende Dienste entschieden:

- **Samba-Dateiserver**

Samba ermöglicht die reibungslose Freigabe von Dateien und Drucken in Netzwerken. Dies erleichtert den plattformübergreifenden Austausch.

Gründe:

- Reibungslose Integration in Windows-Netzwerke, aber auch Zugriff von Linux und macOS aus möglich.
- Bereitstellung von Dateifreigaben für den plattformübergreifenden Austausch von Dateien und die gemeinsame Nutzung von Ressourcen.
- Einfache Konfiguration und Verwaltung von Dateien, was es ideal für den Einsatz in einem Bildungsumfeld wie einem Netzwerk- und Betriebssystemkurs macht.

- **Tomcat-Webserver:** Apache Tomcat ist ein leistungsstarker Webserver. Die Verwendung von Docker ermöglicht eine effiziente Isolierung und Bereitstellung von Tomcat auf dem Raspberry Pi.

Gründe:

- Effiziente und unkomplizierte Bereitstellung mit Docker, um Websites zu hosten.
- Flexibilität und Skalierbarkeit, wenn verschiedene Websites (oder auch Java-Anwendungen, falls benötigt) in unterschiedlichen Entwicklungsstadien auf dem Raspberry Pi ausgeführt werden sollen.

- **Jupyter Notebook:** Jupyter bietet eine interaktive und browserbasierte Plattform für die Entwicklung, Dokumentation und Ausführung von Code in verschiedenen Programmiersprachen. Die Docker-basierte Einrichtung erleichtert die konsistente Bereitstellung eines Jupyter Notebooks, was flexiblen Zugriff auf Daten und Skripte von überall aus ermöglicht.

Gründe:

- Interaktive und browserbasierte Entwicklungsumgebung
- Konsistente Bereitstellung mit Docker, was eine zuverlässige und gut dokumentierte Umgebung schafft, um mit Daten und Skripten zu arbeiten.
- Flexibler Zugriff von überall aus, was die Zusammenarbeit erleichtert.

Diese Anleitung führt durch den gesamten Prozess der Docker- und Docker-Compose-Installation auf dem Raspberry Pi. Anschliessend werden die spezifischen Konfigurationsschritte für jeden Dienst detailliert erläutert. Die Struktur ermöglicht auch Anfängern im Bereich Netzwerk- und Betriebssysteme einen problemlosen Einstieg.

Installation von Docker auf dem Raspberry Pi

Installation und Updates

Bevor wir uns in die Einrichtung von Docker vertiefen, ist es essenziell sicherzustellen, dass das Betriebssystem des Raspberry Pi auf dem neuesten Stand ist. In diesem Beispiel verwenden wir die Linux-Distribution Debian (Version 12.0 - "Bookworm"). Wir führen die folgenden Befehle aus, um das System zu aktualisieren:

```
sudo apt update
sudo apt upgrade -y
```

Diese Befehle sorgen dafür, dass alle verfügbaren Pakete auf den neuesten Stand gebracht werden, um einen reibungslosen Installationsprozess von Docker zu gewährleisten. Um die Kompatibilität mit den neuesten Docker-Features zu garantieren, muss sichergestellt werden, dass der Raspberry Pi mit einer möglichst aktuellen Linux-Distribution arbeitet.

Vorbereitung zur Installation von Docker

Nachdem das Betriebssystem aktualisiert wurde, wird die Installation von Docker fortgesetzt. Hierbei orientiert man sich an den Anweisungen auf der offiziellen Docker-Website, insbesondere für die Ubuntu-Distribution, da Ubuntu auf Debian basiert («Derivat»).

Folgende Schritte beschreiben den Installationsprozess:

(siehe auch <https://docs.docker.com/engine/install/ubuntu/>):

```
# Add Docker's official GPG key:

sudo apt-get update

sudo apt-get install ca-certificates curl gnupg

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg

sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:

echo \

"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \

"${(. /etc/os-release && echo "$VERSION_CODENAME")}" stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update
```

Nach Abschluss dieser Schritte sind alle Voraussetzungen erfüllt, um Docker auf dem Raspberry Pi zu installieren.

Installation von Docker

Für die Vereinfachung der Verwaltung von Multi-Container-Docker-Anwendungen kommt Docker Compose zum Einsatz. Die Installation von Docker sowie des Docker-Compose-Plugins erfolgt durch die Ausführung des folgenden Befehls:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin docker-compose
```

Dieser Befehl gewährleistet nicht nur die Installation von Docker, sondern stellt auch sicher, dass die erforderlichen Komponenten, darunter docker-ce, docker-ce-cli, containerd.io, docker-buildx-plugin und docker-compose-plugin, ordnungsgemäss integriert sind.

Die Verwendung von Docker Compose ermöglicht eine effiziente Definition und Konfiguration der verschiedenen Dienste, die in einer Anwendung arbeiten. Dies ist besonders relevant für unsere Einrichtung, da wir Samba, Tomcat und Jupyter als separate Container-Dienste betreiben werden.

Verwendung von Docker Compose

Um die nachfolgend definierten Dienste zu starten, wird folgender Befehl verwendet. Dieser Befehl startet alle Dienste im Hintergrund (-d):

```
docker-compose up -d
```

Wurden Änderungen an einem lokalen, selbst erstellten, Dockerfile vorgenommen, welches wir in unserem Fall für den Samba-Server benötigen, kann mit folgendem Befehl ein erneutes Bauen des Images aus dem Dockerfile forciert werden:

```
docker-compose up --build --force-recreate
```

Um die Container zu stoppen und alle zugehörigen Ressourcen zu entfernen, wird folgender Befehl verwendet:

```
docker-compose down -v
```

Dadurch werden sämtliche Container gestoppt und alle damit verbundenen Ressourcen aufgeräumt. Dieser Schritt wird z.B. benötigt, wenn Änderungen an der Konfiguration vorgenommen wurden und die Container neu erstellt werden sollen. Wird der Parameter -v weggelassen, bleiben die Volumes bestehen und nur die Container werden entfernt, unsere Daten bleiben also erhalten (Persistent).

Docker Compose Konfiguration

Die ``docker-compose.yml``-Datei definiert drei Hauptdienste: ``samba``, ``tomcat`` und ``jupyter``.

Dienste

- **Samba:**
 - Verwendet ein selbst erstelltes Image zur Bereitstellung des Samba-Dienstes. Das Image verwendet ein eigens geschriebenes Entrypoint Shellsript zur Konfiguration.
 - Erstellt Benutzer, welche in der `.env`-Datei definiert wurden und erstellt für jeden Benutzer ein «Home»-Verzeichnis, sofern die Variable `«CREATE_USER_HOMES»` wahr ist.
 - Erstellt eine allen Benutzern zugängliche Freigabe (`dateiablage`).
 - Ein Volume ermöglicht die Datenpersistenz über alle Freigaben hinweg.
 - Ports `1390` und `4450` sind für den Zugriff auf Samba vom Host aus freigegeben, um Standardport-Kollisionen zu vermeiden.
 - Eine Konfigurationsdatei (`smb.conf`) wird verwendet, um benutzerdefinierte Einstellungen für Samba zu spezifizieren.
- **Tomcat:**
 - Nutzt das neueste offizielle Tomcat-Image von Docker Hub.
 - Port `8080` ist für den Webzugriff vom Host aus freigegeben.
 - Ein Volume speichert die Daten von Tomcat persistent. Im Umfang unseres Projektes ist dies noch nicht zwingend erforderlich, da die Landing-Page separat eingebunden wird (s.u.), für komplexere Anwendungen wird dies jedoch im Regelfall benötigt.
 - Eine lokale `index.html`-Datei wird in den `webapps/ROOT`-Ordner des Tomcat-Servers gemappt, um eine Homepage bereitzustellen.
 - Um die Sicherheit zu erhöhen, könnten zukünftig z.B. auch Zugriffssteuerungs-Mechanismen (z.B., Benutzername/Passwort) konfiguriert werden.
- **Jupyter:**
 - Setzt das `quay.io/jupyter/scipy-notebook`-Image für ein Scipy-Notebook (Scientific Python) ein.
 - Port `8888` ist für den Zugriff auf die Jupyter Notebook-Weboberfläche vom Host aus freigegeben.
 - Ein Volume ermöglicht die persistente Speicherung der Jupyter Notebooks.
 - Ein optionales Token oder Passwort kann für den Zugriff auf die Jupyter-Weboberfläche konfiguriert werden, um die Sicherheit zu erhöhen.
 - Zusätzliche Bibliotheken können durch Erweiterung des Docker-Images oder durch die Verwendung von Jupyter Notebooks oder einer Anpassung der `requirements.txt` hinzugefügt werden.

Netzwerke

- **internal_network:**
 - Ein internes Netzwerk, das die Kommunikation zwischen den Diensten ohne externen Zugang ermöglicht.
- **external_network:**
 - Ein Netzwerk für den Zugriff von aussen sowie vom Host-System.

Volumes

- **samba_volume:**
 - Stellt die Persistenz für den Samba-Dateiserver sicher.
- **tomcat_volume:**
 - Hält die Daten von Tomcat persistent vor.
- **jupyter_volume:**
 - Sichert die Scipy Notebooks des Jupyter-Notebook-Servers dauerhaft.

Zusammenfassung

Diese sorgfältig konfigurierte Docker-Compose-Umgebung eröffnet die Möglichkeit, auf dem Raspberry Pi eine Vielzahl von Netzwerkdiensten einzurichten und zu verwalten. Die klare Definition der Dienste (Samba, Tomcat, Jupyter), Netzwerke und Volumes schafft eine praktische Plattform für die Entwicklung, Bereitstellung und Verwaltung von Anwendungen und Diensten.

Die Vorteile dieser Konfiguration umfassen:

- **Effiziente Ressourcennutzung:** Durch die Verwendung von Containern und der klaren Trennung der Dienste werden Ressourcen effizient genutzt, wodurch ein reibungsloser Betrieb unterschiedlicher Anwendungen auf dem Raspberry Pi ermöglicht wird.
- **Flexibilität und Skalierbarkeit:** Die klare Struktur der Docker-Compose-Konfiguration erlaubt es, Dienste einfach hinzuzufügen, zu entfernen oder zu modifizieren, um den sich ändernden Anforderungen gerecht zu werden. Dies ermöglicht eine einfache Skalierung je nach Bedarf.
- **Persistente Datenspeicherung:** Die Verwendung von Volumes gewährleistet die persistente Speicherung von Daten für Samba, Tomcat und Jupyter. Dies ermöglicht eine dauerhafte Datenspeicherung über Neustarts und Aktualisierungen hinweg.
- **Einfache Verwaltung:** Die einheitliche Verwaltung von Diensten über Docker Compose erleichtert die Handhabung von Start- und Stop-Befehlen sowie das saubere Entfernen von Ressourcen. Dies erleichtert die Wartung und Aktualisierungen.

Insgesamt stellt diese Docker-Compose-Konfiguration eine robuste Basis für die Umsetzung verschiedener Anwendungsfälle dar und unterstützt sowohl Anfänger als auch fortgeschrittene Benutzer bei der Implementierung von Netzwerkdiensten auf einem Raspberry Pi.

Nutzung der Dienste

Samba-Server

- **Verbindungsaufbau über SMB zum Host:**
 - Verwenden über die angepassten Ports 1390 und 4450.
 - Beispiel: \\<Raspberry-Pi-Adresse>:4450 (Windows) oder smb://<Raspberry-Pi-Adresse>:4450 (macOS, Linux)
- **Zugriff auf die Freigabe:**
 - Die Freigabe `dateiablage` sowie die Home-Verzeichnisse sind nur mit einem in der `.env` Datei definierten Benutzernamen und dem dazugehörigen Passwort zugänglich.

Tomcat-Webserver

- **Zugriff auf Tomcat:**
 - Öffnen via Webbrowser und navigieren auf `http://<Raspberry-Pi-Adresse>:8080`.
- **Startseite:**
 - Die `index.html`-Datei im Verzeichnis `/tomcat` dient als Startseite des Tomcat-Servers.

Jupyter Notebook-Server

- **Zugang zu Jupyter Notebook:**
 - Öffnen via Webbrowser und navigieren auf `http://<Raspberry-Pi-Adresse>:8888`
- **Persistenz der Notebooks:**
 - Die Jupyter Notebooks werden im `jupyter_volume` gespeichert und sind dort persistent hinterlegt.

Code & Dokumentation

Der Code sowie diese Dokumentation sind auf Github einsehbar:

<https://github.com/pburkhalter/TEKO-NET-Docker>