Raspberry Pi 4 (2GB RAM) – Heimserver CoreStack

Ziel: Stabiler Betrieb mit **DokuWiki**, **File Browser**, **Cockpit** und **Gogs** (Git) auf einem Raspberry Pi 4 mit **2GB RAM**. **Wichtig:** Alle Befehle sind **copy-paste-fertig**. Die Docker-Stacks sind so aufgebaut, dass sie **heute Abend** direkt getestet werden können. **Fix enthalten: Stack 02 (File Browser) → DB-Datei wird manuell angelegt** (sonst Start-/Login-Probleme).

---

1) Terminal-Cheatsheet (Copy & Paste)

Alles hier ist so geschrieben, dass du es **1:1 ins Terminal kopieren** kannst. Unter jedem Block steht kurz, was er macht.

1.1 System-Check (Architektur & RAM)

```
getconf LONG_BIT
```

Ausgabe: 64 → Dein System läuft in **64-Bit** (wichtig für Docker-Images).

```
free -h
```

Zeigt freien RAM an (ideal: möglichst viel frei vor dem Start).

```
df -h
```

Zeigt freien Speicherplatz (mind. 10GB frei auf / empfohlen).

1.2 System aktualisieren

```
sudo apt update && sudo apt full-upgrade -y
sudo reboot
```

Aktualisiert alle Pakete und startet neu (wichtig nach Kernel-Updates).

Optional (nur Raspberry Pi OS):

```
sudo raspi-config
```

→ **Advanced Options → Expand Filesystem** (nutze die volle SSD/SD-Größe).

1.3 Basis-Tools installieren

```
sudo apt install -y mc git curl wget ca-certificates gnupg
```

- **mc** – Midnight Commander (Dateimanager)
- **git** – für Repos/Tools
- **curl/wget** – Download-Tools
- **gnupg/ca-certificates** – für Docker-Repo

---

2) Digitalcraft Ordnerstruktur (einheitlich)

Root: `/srv/docker/corestack` **Pro Dienst ein Ordner:**

- `/srv/docker/corestack/wikicore` (DokuWiki)
- `/srv/docker/corestack/filebrowser` (File Browser)
- `/srv/docker/corestack/gogs` (Git-Repository)
- `/srv/docker/corestack/shared` (Gemeinsame Daten)

2.1 Ordner anlegen

```
sudo mkdir -p /srv/docker/corestack/{wikicore,filebrowser,gogs,shared}
sudo chown -R $USER:$USER /srv/docker/corestack
```

Erklärung:

- `mkdir -p` erstellt alle Ordner inkl. Zwischenverzeichnisse
- chown setzt dich als Besitzer (im CoreStack weniger `sudo` nötig)

2.2 In den Stack-Root wechseln

```
cd /srv/docker/corestack
```

Ab hier arbeitest du im CoreStack-Root.

---

3) Cockpit installieren (Host-System)

```
sudo apt install -y cockpit
sudo systemctl enable --now cockpit.socket
```

Zugriff:

```
https://<PI-IP>:9090
```

Nutze Cockpit für:

- System-Logs
- Dienste verwalten
- Updates
- Speicherplatz überwachen

---

4) Docker + Compose installieren (Debian/Raspberry Pi OS)

4.1 Docker GPG-Key speichern

```
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --
dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

4.2 Docker Repo eintragen

```
echo "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update
```

4.3 Docker installieren

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

4.4 Docker ohne sudo nutzen

```
sudo usermod -aG docker $USER
newgrp docker
```

```
docker version
docker compose version
```

Test:

```
docker run --rm hello-world
```

→ Sollte „Hello from Docker!“ ausgeben.

4.5 Gemeinsames Docker-Netzwerk erstellen

```
docker network create dc-net
docker network ls | grep dc-net
```

Zweck: Alle Container kommunizieren über `dc-net` (z. B. `dc-gogs` statt IP-Adressen).

---

5) Docker-Stacks (RAM-optimiert & funktionssicher)

Wichtig: RAM-Limits nutzen wir mit `mem_limit` – das wirkt in normalem Docker Compose zuverlässig. (Hinweis: `deploy.resources` wird außerhalb von Docker Swarm oft ignoriert.)

Stack 01 – DokuWiki (WikiCore)

Pfad: `/srv/docker/corestack/wikicore/`

1) .env

Datei: `/srv/docker/corestack/wikicore/.env`

```
TZ=Europe/Berlin
PUID=1000
PGID=1000
WIKICORE_PORT=8080
```

Hinweis: `PUID/PGID` prüfen:

```
id -u
id -g
```

2) docker-compose.yml

Datei: `/srv/docker/corestack/wikicore/docker-compose.yml`

```
services:
  wikicore:
    image: lscr.io/linuxserver/dokuwiki:latest
    container_name: dc-wikicore
    environment:
      - TZ=${TZ}
      - PUID=${PUID}
      - PGID=${PGID}
    volumes:
      - ./config:/config
    ports:
      - "${WIKICORE_PORT}:80"
    networks:
      - dc-net
    restart: unless-stopped
    mem_limit: 256m

networks:
  dc-net:
    external: true
```

3) Start

```
cd /srv/docker/corestack/wikicore
docker compose up -d
docker compose ps
```

Zugriff:

```
http://<PI-IP>:8080
```

Stack 02 – File Browser (KORRIGIERT: DB-Datei manuell anlegen!)

Pfad: `/srv/docker/corestack/filebrowser/`

Warum dieser Fix wichtig ist: File Browser nutzt eine SQLite-DB. Wenn die DB-Datei/der DB-Ordner nicht existiert oder nicht beschreibbar ist, startet der Container zwar ggf. – aber Settings/Login/Initialisierung gehen schief. **Lösung:** DB-Datei einmalig anlegen + Rechte sauber setzen.

1) Vorbereitung (Ordner + DB-Datei + Rechte)

```
cd /srv/docker/corestack/filebrowser

# Ordnerstruktur anlegen (config + database sind Pflicht)
mkdir -p config database

# DB-Datei manuell anlegen (WICHTIG!)
touch database/filebrowser.db

# Rechte setzen (Standard-User 1000:1000)
sudo chown -R 1000:1000 config database
```

2) .env

Datei: `/srv/docker/corestack/filebrowser/.env`

```
TZ=Europe/Berlin
PUID=1000
PGID=1000
FILEBROWSER_PORT=8081
```

3) docker-compose.yml (robust + eindeutig)

Datei: `/srv/docker/corestack/filebrowser/docker-compose.yml`

```
services:
  filebrowser:
    image: filebrowser/filebrowser:latest
    container_name: dc-filebrowser
    user: "${PUID}:${PGID}"
    ports:
      - "${FILEBROWSER_PORT}:80"
    volumes:
      - /srv/docker/corestack/shared:/srv
      - ./config:/config
      - ./database:/database
    command: >
      --database /database/filebrowser.db
      --root /srv
      --port 80
    networks:
      - dc-net
    restart: unless-stopped
    mem_limit: 128m
```

```
networks:
  dc-net:
    external: true
```

4) Start

```
cd /srv/docker/corestack/filebrowser
docker compose up -d
docker compose ps
```

Zugriff:

```
http://<PI-IP>:8081
```

Standard-Login (bei neuer DB):

```
Username: admin
Password: admin
```

Tipp: Direkt nach dem ersten Login Passwort ändern.

Stack 03 – Gogs (Git-Repository)

Pfad: `/srv/docker/corestack/gogs/`

1) .env

Datei: `/srv/docker/corestack/gogs/.env`

```
TZ=Europe/Berlin
GOGS_HTTP_PORT=3000
GOGS_SSH_PORT=2222
```

2) docker-compose.yml

Datei: `/srv/docker/corestack/gogs/docker-compose.yml`

```
services:
  gogs:
    image: gogs/gogs:latest
    container_name: dc-gogs
```

```
    environment:
      - TZ=${TZ}
    volumes:
      - ./data:/data
    ports:
      - "${GOGS_HTTP_PORT}:3000"
      - "${GOGS_SSH_PORT}:22"
    networks:
      - dc-net
    restart: unless-stopped
    mem_limit: 256m

networks:
  dc-net:
    external: true
```

3) Start

```
cd /srv/docker/corestack/gogs
docker compose up -d
docker compose ps
```

4) Gogs konfigurieren

```
1. Öffne ''http://<PI-IP>:3000''
2. **Datenbank:** Wähle **SQLite3** (kein PostgreSQL nötig)
3. **Domain:** ''<PI-IP>'' (oder Hostname, falls DNS)
4. **SSH-Port:** ''2222'' (falls Git über SSH)
5. Admin-Benutzer erstellen und speichern
```

Git SSH:

```
ssh -p 2222 git@<PI-IP>
```

---

6) Start-Reihenfolge (empfohlen)

```
# 1) DokuWiki
cd /srv/docker/corestack/wikicore && docker compose up -d

# 2) File Browser (inkl. DB-Fix vorher!)
cd /srv/docker/corestack/filebrowser && docker compose up -d

# 3) Gogs
```

```
cd /srv/docker/corestack/gogs && docker compose up -d
```

---

7) Update-Routine (pro Stack)

```
cd /srv/docker/corestack/<stack>
docker compose pull
docker compose up -d
docker image prune -f
```

---

8) Backup-Quicklist (Pflicht)

Wichtige Pfade sichern:

- `/srv/docker/corestack/wikicore/config/` (DokuWiki)
- `/srv/docker/corestack/gogs/data/` (Gogs Repos + Config)
- `/srv/docker/corestack/filebrowser/config/` (File Browser Settings)
- `/srv/docker/corestack/filebrowser/database/` (File Browser DB!)
- `/srv/docker/corestack/shared/` (Gemeinsame Daten)

Backup-Befehl (Beispiel):

```
cd /srv/docker/corestack
tar -czvf backup_$(date +%Y-%m-%d).tar.gz wikicore/config gogs/data
filebrowser/config filebrowser/database shared/
```

---

9) Dienste testen (Browser)

Ersetze <PI-IP> durch die IP deines Raspberry Pi:

Dienst	URL	Port
Gogs	http://<PI-IP>:3000	3000
DokuWiki	http://<PI-IP>:8080	8080
File Browser	http://<PI-IP>:8081	8081
Cockpit	https://<PI-IP>:9090	9090

---

10) RAM-Überwachung (praxisnah)

```
sudo apt install -y htop
```

```
htop
```

Docker live prüfen:

```
docker stats
```

Ziel: Unter Last möglichst **unter ~1.8GB RAM** bleiben (sonst Swapping → Performance-Einbruch).

---

11) Fehlerdiagnose (wenn etwas nicht startet)

```
# Status
docker ps -a

# Logs (Beispiel)
docker logs --tail 120 dc-filebrowser
docker logs --tail 120 dc-wikicore
docker logs --tail 120 dc-gogs

# Ports belegt?
sudo ss -tulpn | grep -E ':8080|:8081|:3000|:2222|:9090'

# Rechte prüfen (File Browser)
ls -la /srv/docker/corestack/filebrowser
ls -la /srv/docker/corestack/filebrowser/database
```

File Browser DB-Fix (wenn es hakt):

- Existiert `database/filebrowser.db`?
- Ist `database/` beschreibbar für UID/GID (1000:1000)?
- Läuft der Container mit `user: „1000:1000“`?

Fertig! Dein **Raspberry Pi 4 (2GB RAM)** läuft jetzt stabil mit:

- **DokuWiki** (Dokumentation)
- **File Browser** (Dateimanagement)
- **Gogs** (Git-Repository)
- **Cockpit** (Systemverwaltung)

Viel Erfolg beim Test heute Abend – damit der Videodreh morgen absolut sauber läuft.