Portainer Housekeeping

Dieser Artikel beschreibt die regelmäßige Wartung und Fehleranalyse für **Portainer**.
Ziel: Tote Endpoints vermeiden, Logs im Griff behalten, Plattenplatz sichern und stabile Funktion sicherstellen.

1. Environments prüfen

Portainer verwaltet Umgebungen („Environments“) → jede zeigt auf einen Docker-Socket oder eine Remote-API.

- Menü: **Environments**
  - Prüfen, ob „Local“ vorhanden ist:
  - Typ: **Local**

    ```
    * Endpoint: `unix:///var/run/docker.sock`
    ```

    * Tote Einträge (z. B. alte Hosts mit IP:2376) → **löschen** oder **deaktivieren**

  - Tipp: sprechende Namen vergeben („docker-mash“, „cluster-node1“)

CLI-Check

<code bash> # Test: Local Docker-Socket erreichbar? curl –unix-socket /var/run/docker.sock http://localhost/_ping # OK → pong </code>

2. Docker-Socket korrekt mounten

In `docker-compose.yml` für Portainer muss enthalten sein:

```
volumes:
  - /var/run/docker.sock:/var/run/docker.sock
  - portainer_data:/data
```

→ ohne diesen Mount funktioniert „Local environment“ nicht.

3. Snapshots und Polling

Portainer pingt die Environments regelmäßig (Container-Snapshot, Healthcheck).

- Menü: **Settings → Snapshots**
  - Intervalle anpassen (z. B. 5–15 min statt 1 min)

* Snapshot für inaktive/tote Environments deaktivieren

4. Logs im Blick

Vollgelaufene Container-Logs führen zu „Platte 100 %“ und Fehlermeldungen.

Container-Logs begrenzen

Im Compose jedes Dienstes:

```
logging:
  driver: json-file
  options:
    max-size: "50m"
    max-file: "5"
```

Offene gelöschte Dateien prüfen

```
lsof +L1 | head -n 20
```

→ zeigt Prozesse mit „gelöschten, aber noch offenen“ Dateien.

5. Disk Usage überwachen

```
df -h /var/lib/docker /var/run
df -i /var/lib/docker        # Inodes
docker system df
docker system df -v
```

6. Typische Fehlerbilder

- **„Failed to find local environment“**

  ```
  → Local Socket nicht gemountet / falsche Rechte / Environment gelöscht
  ```

  * **„connect: connection refused 193.197.168.21:2376“**

  ```
  → Remote Docker-API down oder Host nicht erreichbar
  ```

  * **„no route to host“**

```
→ Firewall/VPN/Netzwerkproblem
```

* **„cannot find webhook by containerId“**

```
→ Container entfernt, DB-Eintrag verwaist → Stack neu deployen
```

7. Aufräumen & Pflege

Regelmäßig:

```
# unbenutzte Images, Container, Netzwerke, Volumes
docker system prune -af --volumes
```

- Vorher prüfen: **Achtung** löscht auch gestoppte Container und nicht genutzte Volumes.

8. Absicherung

- Portainer selbst hinter **Traefik/HTTPS** betreiben
  * Zugriff nur aus internen Netzen oder via VPN
  * Admin-Account mit 2FA absichern

9. Backup

- Persistente Daten liegen in `portainer_data`
  * Backup reicht: Volume oder `/opt/stacks/portainer/` + Daten sichern
- Restore: neues Portainer-Container mit altem Volume starten

10. Checkliste „Fehlerfall“

1. [ ] Ist die Platte voll? (`df -h`)
   - [ ] Sind alte Environments noch eingetragen?
   - [ ] Ist der Docker-Socket gemountet?
   - [ ] Funktioniert `curl --unix-socket ... _ping`?
   - [ ] Snapshot-Intervall zu klein?
   - [ ] Offene gelöschte Logs (`lsof +L1`)?