Docker Überwachung und Sicherheit

Arbeitsspeicher und Swap Auslastung

```
free -lh
```

Festplattenauslastung der Docker Container

```
docker system df
```

Informationen über den Docker Host

```
docker system info
```

Laufende Container gegen Fehlkonfigurationen prüfen

Docker Bench for Security ist ein Skript, das dein gesamtes Docker-Setup anhand der Empfehlungen des Center for Internet Security (CIS) auditiert. Es prüft z.B. Host-Sicherheitskonfiguration, Container-Einstellungen und Netzwerkregeln:
Installation und Aiufrug

```
git clone https://github.com/docker/docker-bench-security.git
cd docker-bench-security
sudo sh docker-bench-security.sh
```

Das Skript erstellt einen detaillierten Bericht mit Verbesserungsvorschlägen

Logs anzeigen

```
docker logs CONTAINERNAME
```

Eine Liste der Commands findest du in der Docker Doku

Passwörter in docker-compose files

Um Passwörter in Docker Compose **nicht im env-File** zu hinterlegen, sondern als Datei-gehaltenes Secret zu verwenden, nutzt du die Compose-„secrets"-Funktion. Diese stellt Daten im Container unter /run/secrets/<name> als Datei bereit. Das offizielle Nextcloud-Image (und die meisten aktuellen Datenbank-Container) unterstützen die Übergabe sensibler Daten per *_FILE-Umgebungsvariablen, die auf diese Dateien zeigen.

Ein exemplarisches docker-compose.yml für Nextcloud & Postgres mit Secret-Verwendung könnte so aussehen:

```
version: '3.9'

services:
  db:
    image: postgres:16
    restart: always
    volumes:
      - db_data:/var/lib/postgresql/data
    environment:
      POSTGRES_DB_FILE: /run/secrets/postgres_db
      POSTGRES_USER_FILE: /run/secrets/postgres_user
      POSTGRES_PASSWORD_FILE: /run/secrets/postgres_password
    secrets:
      - postgres_db
      - postgres_user
      - postgres_password

  app:
    image: nextcloud:latest
    restart: always
    ports:
      - 8080:80
    volumes:
      - nextcloud_data:/var/www/html
    environment:
      POSTGRES_HOST: db
      POSTGRES_DB_FILE: /run/secrets/postgres_db
      POSTGRES_USER_FILE: /run/secrets/postgres_user
      POSTGRES_PASSWORD_FILE: /run/secrets/postgres_password
      NEXTCLOUD_ADMIN_USER_FILE: /run/secrets/nc_admin_user
      NEXTCLOUD_ADMIN_PASSWORD_FILE: /run/secrets/nc_admin_pw
    depends_on:
      - db
    secrets:
      - postgres_db
      - postgres_user
      - postgres_password
      - nc_admin_user
      - nc_admin_pw

volumes:
  db_data:
  nextcloud_data:

secrets:
  postgres_db:
    file: ./secrets/postgres_db.txt
  postgres_user:
```

```
    file: ./secrets/postgres_user.txt
  postgres_password:
    file: ./secrets/postgres_password.txt
  nc_admin_user:
    file: ./secrets/nc_admin_user.txt
  nc_admin_pw:
    file: ./secrets/nc_admin_pw.txt
```

Rechte des ./secrets file (Beispiel)

```
sudo chown root:33 /directory/secrets/smtp_password
sudo chmod 0440 /directory/secrets/smtp_password
```

Wichtige Hinweise:

- Die Secret-Dateien müssen als Plaintext (ohne Anführungszeichen/Leerzeilen) z.B. in `./secrets/nc_admin_pw.txt` liegen.
- Die *_FILE-Umgebungsvariablen müssen (*statt* der normalen Variablen) genutzt werden, z.B. `NEXTCLOUD_ADMIN_PASSWORD_FILE` statt `NEXTCLOUD_ADMIN_PASSWORD`.
- Diese Technik wird von Compose ab v3.1+ (nicht nur im Swarm) unterstützt, und ist für Nextcloud/Postgres-Images dokumentiert.
- Alle Secrets werden als Datei **nur** im Container bereitgestellt und tauchen nicht in Umgebungsvariablen oder Logs auf.

So bleiben sensible Informationen aus der Compose-Datei, dem ENV-File und sämtlichen Logfiles heraus und werden sicher als Datei injiziert.

Resourcenauslastung jedes einzelnen laufenden Docker Container

```
docker stats --no-stream
```