

Docker Security

Modul 169

Inhalt

- **Docker Security**
- **Übungen**
zu Docker Security
- **Repetition**
- **An Aufgaben der letzten Wochen arbeiten**
- **Übungstest auf Classtime**

Regeln

§1 Fokus und Geräte

Die **digitalen Geräte**:  ,  , etc.

- immer nur auf **Aufforderung der Lehrkraft**
- immer nur zur **Bearbeitung der gestellten Aufgaben**

Private Aktivitäten sind untersagt: *unter anderem Social Media, Spiele, Videos, private E-Mails/Chats, Surfen, Shoppen, etc.*

§2 Ruhe und Umgangsformen

Die Konzentration der Mitschüler muss gewährleistet sein.

- **Lärm ist zu vermeiden**
z.B. laute Gespräche, Geräusche, Rufen.
- **Freundlicher, höflicher und respektvoller Umgangston**

Wofür werden Docker Volumen verwendet?

Worin unterscheiden sich Volumen zu Bind Mounts?

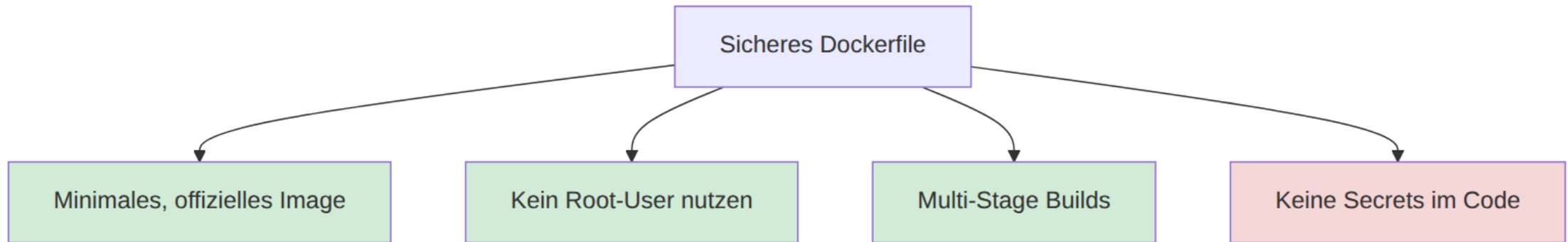
Auftrag

Machen Sie beliebige Übungen den vorherigen Wochen.

- Übungen Mounts
- Übungen Volumes

-  Einzelarbeit
-  bis zur grossen Pause

Sicherheitsprinzipien im Dockerfile



Minimales, offizielles Image

Minimal

```
FROM node:20-alpine
WORKDIR /app
COPY package*.json ./
RUN npm install --production
COPY . .
CMD ["node", "index.js"]
```

💡 Weniger Programme, weniger Angriffsfläche.

Ubuntu mit node installiert

```
FROM ubuntu:22.04
RUN apt-get update && apt-get install -y \
    curl \
    gnupg \
    && curl -fsSL https://deb.nodesource.com | bash - \
    && apt-get install -y nodejs && apt-get clean \
    && rm -rf /var/lib/apt/lists/*
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
CMD ["node", "index.js"]
```

Kein Root-User nutzen

```
FROM ubuntu:22.04
RUN apt-get update && apt-get install -y curl \
    && curl -fsSL https://deb.nodesource.com | bash - \
    && apt-get install -y nodejs && rm -rf /var/lib/apt/lists/*
# Eigenen Benutzer und Gruppe anlegen und verwenden
RUN adduser --system --group --home /home/appuser appuser
WORKDIR /app
RUN chown appuser:appuser /app # Rechte setzen
USER appuser # User appuser verwenden
COPY --chown=appuser:appuser package*.json ./ # Berechtigungen geben
RUN npm install --production
COPY --chown=appuser:appuser . .
CMD ["node", "index.js"] # App starten
```

- 💡 Weniger Rechte ist immer sicherer!

Minimale Images haben oft einen User parat

```
FROM node:20-alpine

# Verzeichnis erstellen und Berechtigungen setzen
RUN mkdir -p /home/node/app && chown -R node:node /home/node/app
WORKDIR /home/node/app

# Zum Nicht-Root-Benutzer wechseln
USER node

# Dateien kopieren und dabei direkt den Besitzer ändern (--chown)
COPY --chown=node:node package*.json ./
RUN npm install

COPY --chown=node:node . .

CMD ["node", "index.js"]
```

Multistage verkleinert produktives Image

```
FROM node:20 AS builder
WORKDIR /app
COPY package*.json ./
COPY server.js ./
RUN npm install

# Stage 2: Production stage "slim!"
FROM node:20-slim
WORKDIR /app
COPY --from=builder /app .
EXPOSE 3000
CMD ["npm", "start"]
```

- 💡 Weniger Programme, weniger Angriffsfläche.

Keine Secrets im Code

✓ Richtig

```
# Datei mit Secret erstellen, Achtung: Bash-History leeren!  
echo "MY_PASSWORD=super-geheim-123" > .env.secret  
  
# Container starten und die Datei einbinden  
docker run -d \  
  --name meine-app \  
  --env-file .env.secret \  
  mein-node-image
```

- Secrets werden als ENV-Vars in den **Container beim Starten** geladen.
- `*.secret` ins `.gitignore`.

✗ Falsch

```
FROM node:20-alpine  
# ✗ PIIIP Falsch!  
ENV MY_PASSWORD="super-geheim-123"
```

- **NIE** als ENV im Dockerfile
- **NIE** in git eingcheckedt!



Supersicher → fnox

```
fnox init
# Ein Secret verschlüsselt hinzufügen (wird in fnox.toml gespeichert)
fnox set DB_PASSWORD "mein-super-geheimes-passwort"

# fnox entschlüsselt DB_PASSWORD und übergibt es an Docker
fnox exec -- docker run -d \
  --name meine-app \
  -e DB_PASSWORD \
  mein-node-image
```

- Dadurch ist das Passwort auch auf der Maschine nicht in Klartext vorhanden und könnte sogar in git eingchecked werden.
- Mehr auf [fnox](#)

Secrets mit docker-compose.yml

```
# Datei mit Secret erstellen, Achtung: Bash-History leeren!  
echo "MY_PASSWORD=super-geheim-123" > password.txt
```

```
services:  
  app:  
    image: mein-node-image  
    secrets:  
      - db_password  
  
secrets:  
  db_password:  
    file: ./password.txt
```

Auftrag

Lesen Sie "Docker Security" von der Woche 4

- [Docker Security](#)

-  Einzelarbeit
-  15 min



Auftrag

Gehen Sie durch alle `Dockerfile` Aufgaben von Woche 2 durch und versuchen Sie mit KI Ihrer Wahl die Aufgaben sicherer zu machen.

- 🧬 Einzelarbeit
- ⌚ 15 min

Auftrag

Lösen Sie die **Übungen der Woche 4**.

-  Einzelarbeit
-  45 min