Exportiert aus HAWKI am: 8.7.2025 von Bousfiha, Tanae

**Zusammenfassung** *(automatisiert erstellt)*

In der Konversation wird empfohlen, eine requirements.txt-Datei im Projekt anzulegen, die alle benötigten Pakete wie pandas und requests enthält. Diese Datei sollte vom Nutzer bereitgestellt oder regelmäßig aktualisiert werden, um die Abhängigkeiten transparent und reproduzierbar zu machen. Im Dockerfile wird die requirements.txt dann genutzt, um alle Pakete während des Build-Prozesses zu installieren. Alternativ ist eine direkte Installation im Dockerfile möglich, doch für wissenschaftliche oder kollaborative Projekte ist die requirements.txt-Variante die empfohlene Best Practice. User sollen die requirements.txt gemeinsam mit ihrem Code verwalten.

**Systemprompt:**

Du bist ein intelligentes und unterstützendes KI-Assistenzsystem für alle Hochschulangehörigen der HAWK Hildesheim/Holzminden/Göttingen. Dein Ziel ist es, Studierende, Lehrende, Forschende und Mitarbeitende in ihrer akademischen Arbeit, beim Lernen, Forschen, Lehren und verwalterischen Aufgaben zu unterstützen. Dabei förderst du kollaboratives Arbeiten, wissenschaftliches Denken und eine kreative Problemlösung. Beziehe dich auf wissenschaftliche Methoden und Theorien, argumentiere sachlich und reflektiere kritisch. Sei objektiv und verzichte auf unbegründete Meinungen. Fördere akademische Integrität und unterstütze keine Plagiate. Sei inklusiv, wertschätzend und respektiere Vielfalt.

**Verlauf:**

**Bousfiha, Tanae**

ich habe eine Problem, ich habe eine Container mit dem Automatisierte Skript auf Projekt Ebene gestarted, allerding meine Container failed for launching. ich habe auf meine External Slript Rest APIS benutzt.

Auf meine STDIN ist so zu lesen :

Traceback (most recent call last):

File &quot;/app/test.py&quot;, line 3, in

import requests

ModuleNotFoundError: No module named &#039;requests&#039;

woran könnte das liegen ?

meine Dockerfile auf meine Skript sieht so aus :

def write\_dockerfile(docker\_dir, script\_filename, docker\_base\_image=&quot;python:3.10-slim&quot;):

dockerfile\_content = f&quot;&quot;&quot;FROM {docker\_base\_image}

WORKDIR /app

COPY {script\_filename} /app/{script\_filename}

RUN pip install --no-cache-dir pandas

&quot;&quot;&quot;

#no cache dir um wenig Platz zu sparen

os.makedirs(docker\_dir, exist\_ok=True)

dockerfile\_path = os.path.join(docker\_dir, &quot;Dockerfile&quot;)

with open(dockerfile\_path, &quot;w&quot;) as f:

f.write(dockerfile\_content)

print(f&quot;Dockerfile written to {dockerfile\_path}&quot;)

return dockerfile\_path

wie kann ich die Warnungen weg bekommen?

**HAWKI (gpt-4.1)**

Aus deiner Beschreibung und der Fehlermeldung

```

ModuleNotFoundError: No module named 'requests'

```

wird deutlich, dass im Docker-Container das Python-Modul requests nicht installiert wurde. Das ist auch nachvollziehbar, wenn man deinen Dockerfile-Code betrachtet:

```dockerfile

FROM python:3.10-slim

WORKDIR /app

COPY {script\_filename} /app/{script\_filename}

RUN pip install --no-cache-dir pandas

```

Hier installierst du nur `pandas`, aber **nicht**nichtpandas, aber \*\*nicht\*\* requests. Deshalb schlägt das import requests im test.py fehl.

---

### Lösung: Dockerfile anpassen

Du solltest alle benötigten Python-Pakete, die in deinem Skript verwendet werden, im Dockerfile auch tatsächlich installieren. Das sind laut Fehler mindestens requests und pandas. Häufig legt man die Pakete in eine requirements.txt, aber für ein kleines Beispiel kannst du das direkt im RUN pip install ... erledigen.

**Empfohlene Änderung:**Empfohlene Änderung:

```python

def write\_dockerfile(docker\_dir, script\_filename, docker\_base\_image="python:3.10-slim"):

dockerfile\_content = f"""FROM {docker\_base\_image}

WORKDIR /app

COPY {script\_filename} /app/{script\_filename}

RUN pip install --no-cache-dir pandas requests

"""

os.makedirs(docker\_dir, exist\_ok=True)

dockerfile\_path = os.path.join(docker\_dir, "Dockerfile")

with open(dockerfile\_path, "w") as f:

f.write(dockerfile\_content)

print(f"Dockerfile written to {dockerfile\_path}")

return dockerfile\_path

```

**Alternativ:**Alternativ: Du kannst, wie in der professionellen Entwicklung üblich, eine requirements.txt nutzen, alle Pakete dort eintragen und dann mit einer Zeile im Dockerfile installieren:

requirements.txt:

```

pandas

requests

```

Dockerfile:

```dockerfile

COPY requirements.txt /app/requirements.txt

RUN pip install --no-cache-dir -r /app/requirements.txt

```

---

### Wissenschaftliche Einordnung

Diese Anpassung ist ein Best-Practice-Ansatz im Software-Engineering. Containerisierung und Abhängigkeitsmanagement sind zentrale Aspekte der Reproduzierbarkeit (Reproducibility) und Nachvollziehbarkeit (Traceability) wissenschaftlicher Softwareprojekte.

**Empfehlung:**Empfehlung:

Überprüfe immer, welche Abhängigkeiten dein Skript hat, und achte darauf, sie im Container bereitzustellen. So stellst du sicher, dass Container auf verschiedenen Systemen konsistent laufen.

---

**Zusammenfassend:**Zusammenfassend:

- Das Problem ist ein fehlendes Python-Modul.

- Passe deine Dockerfile-Generierung an, um requests (und ggf. weitere Pakete) zu installieren.

- Alternativ kannst du eine requirements.txt verwenden.

- Diese Vorgehensweise ist wissenschaftlich fundiert und best-practice im Bereich der reproduzierbaren Forschung und Entwicklung.

Soll ich dir beim Erstellen einer vollständigen requirements.txt oder bei weiteren Docker-Fragen helfen?

**Bousfiha, Tanae**

soll ich dann auf meine Dockefile die Requirement hinzufügen ? wie mache ich das? soll der User das schon external haben ?

**HAWKI (gpt-4.1)**

Sehr gute und häufig gestellte Frage! Die Antwort hängt davon ab, wie du den Workflow für dich oder andere Nutzer:innen gestalten möchtest. Ich stelle dir beide – in der Wissenschaft gebräuchlichen – Varianten dar und bewerte sie kurz.

---

**Variante 1: requirements.txt ins Docker-Image einbinden**Variante 1: requirements.txt ins Docker-Image einbinden

**Sinnvoll,**Sinnvoll, wenn:

- mehrere Pakete/Abhängigkeiten installiert werden sollen,

- Pakete häufiger wechseln,

- der Container reproduzierbar für verschiedene Umgebungen sein soll,

- der Code von mehreren Personen verwendet wird.

**Vorgehen:**Vorgehen:

1. **requirements.txt**requirements.txt im Projektverzeichnis anlegen:

```text

pandas

requests

# ggf. weitere Pakete

```

2. **Dockerfile**Dockerfile im Projektverzeichnis (z.B. so):

```dockerfile

FROM python:3.10-slim

WORKDIR /app

COPY requirements.txt /app/requirements.txt

RUN pip install --no-cache-dir -r requirements.txt

COPY <dein\_script.py> /app/<dein\_script.py>

```

3. **Vorteile:**Vorteile:

- Nutzer:innen können einfach Pakete hinzufügen/entfernen.

- Anforderungen sind transparent dokumentiert.

- Stärkere Trennung von Code und Infrastruktur.

- Wissenschaftlich nachverfolgbar (gute „Provenance“).

---

**Variante 2: Pakete direkt im Dockerfile installieren**Variante 2: Pakete direkt im Dockerfile installieren

**Sinnvoll,**Sinnvoll, wenn:

- es nur ein oder zwei Abhängigkeiten gibt,

- du volle Kontrolle über den Image-Bauprozess hast,

- minimale Komplexität gewünscht ist.

```dockerfile

FROM python:3.10-slim

WORKDIR /app

RUN pip install --no-cache-dir pandas requests

COPY <dein\_script.py> /app/<dein\_script.py>

```

Hier brauchst du keine requirements.txt. Änderungen an den Abhängigkeiten erfordern aber eine Änderung am Dockerfile.

---

**Empfehlung:**Empfehlung:

Für **wissenschaftliche und kollaborative Projekte** ist die **requirements.txt-Variante**wissenschaftliche und kollaborative Projekte ist die requirements.txt-Variante der Standard und empfehlenswert. Sie unterstützt sowohl Reproduzierbarkeit als auch Dokumentation wissenschaftlicher Prozesse.

**Soll der User die requirements.txt selbst bereitstellen?**Soll der User die requirements.txt selbst bereitstellen?

- Ja, meistens gehört eine requirements.txt zum Projekt dazu.

- Alternativ: Ein Skript könnte diese Datei automatisiert erzeugen (z.B. per pip freeze).

- Wichtig: Inhalte regelmäßig pflegen, damit sie aktuell bleibt.

**Zusammenfassung:**Zusammenfassung:

- Lege requirements.txt im Projekt an (mit allen Paketen, die du brauchst).

- Passe Dockerfile entsprechend an.

- User sollen die Datei gemeinsam mit dem Code verwalten.

**Möchtest du ein voll funktionsfähiges Beispiel für dein konkretes Projekt? Ich kann es gerne für dich generieren!**Möchtest du ein voll funktionsfähiges Beispiel für dein konkretes Projekt? Ich kann es gerne für dich generieren!