Zusammenfassung Docker Prüfung 2

Inhalt

[Sie können mit YAML-Dateien für docker-compose korrekt umgehen 1](#_Toc198585583)

[Sie kennen die Syntax von docker-compose.yml und können die Schlüsselwörter korrekt anwenden 2](#_Toc198585584)

[Schlüsselwörter im Detail 3](#_Toc198585585)

[Sie können das Kommando docker compose up/down mit den Schaltern -d und --build korrekt anwenden 3](#_Toc198585586)

[Sie können für Sie unbekannte Images anhand verfügbarer Dokumentationen korrekte docker compose Kompositionen aufbauen 3](#_Toc198585587)

[Sie können Dockerfiles für Multistage Builds im Zusammenhang mit dotnet aufbauen 3](#_Toc198585588)

[Sie können Dockerfiles im Zusammenspiel mit docker compose korrekt verwenden 4](#_Toc198585589)

[Sie können mit dotnet Web- und Konsolen-Anwendungen korrekt umgehen und containerisieren 4](#_Toc198585590)

# Sie können mit [YAML-Dateien](https://moodle.gbssg.ch/mod/page/view.php?id=872) für docker-compose korrekt umgehen

**#** Zeilenkommentar

Text kann mit einfachen oder doppelten Hochkommas ', " geklammert werden. Dies ist allerdings nur nötig, wenn der Text Sonderzeichen, z.B. Escape-Sequenz enthält

Eine Liste wird mit - + Abstand gebildet, z.B.

|  |
| --- |
| # A list of tasty fruits  - Apple  - Orange  - Strawberry  - Mango |

Key-Value-Paare werden mit : + Abstand gebildet, z.B.:

|  |
| --- |
| name: Hans Martin |

Objekte bestehend aus mehreren Key-Value-Paaren, können durch Zeilenumbrüche gebildet werden

|  |
| --- |
| vorname: Hans Martin  nachname: Keller  alter: 20 |

Ein Value kann seinerseits wiederum rekursiv aus Objekten, Key-Value-Paaren oder Listen bestehen:

|  |
| --- |
| *# Employee records*  *- martin:*  *name: Martin D'vloper*  *job: Developer*  *skills:*  *- python*  *- perl*  *- pascal*  *- tabitha:*  *name: Tabitha Bitumen*  *job: Developer*  *skills:*  *- lisp*  *- fortran*  *- erlang* |

Die Strukturierung der Elemente wird durch Einrückungen erreicht. Dabei müssen Leerzeichen verwendet werden. Tabulatoren auch am Zeilenende führen zu Fehlern.

# Sie kennen die Syntax von docker-compose.yml und können die Schlüsselwörter korrekt anwenden

* **services**: Zur Definition der einzelnen Container. Die Dienstnamen (Keys) für die Container können freigewählt werden.
* **networks**: Zur Definition eigener Netzwerke
* **volumes**: Angabe von benannten Volumes
* **secrets**: Angabe von Passwortdateien

|  |
| --- |
| * # Grundsätzlicher Aufbau von docker-compose.yml * services: * dienstname1: * schlüsselwort1: einstellung1 * schlüsselwort2: einstellung2 * ... * dienstname2: * schlüsselwort1: einstellung1 * schlüsselwort2: einstellung2 * ... * volumes: * networks: * secrets: |

Die wichtigsten Schlüsselwörter unterhalb der Dienstnamen sind:

* **image**: Das Basisimage eines Service oder
* **build**: Ein Verzeichnis mit Dockerfile, der Service wird dann nicht aus einem Image gestartet sondern aus einem Dockerfile
* **container\_name**: Der Name für den resultierenden Container
* **restart**: always, on-failure oder unless-stopped
* **environment**: Liste mit Umgebungsvariablen für einen Container
* **volumes**: Liste der gemounteten Volumes
* **ports**: Liste der Portweiterleitungen
* **expose**: Ports für die Kommunikation zwischen Containern
* **networks**: Verweis auf ein im Top-Level-Schlüsselwort networks definiertes Netzwerk
* **secrets**: Verweis auf die im Top-Level-Schlüsselwort secrets definierte Passwortdatei

## Schlüsselwörter im Detail

# Sie können das Kommando [docker compose](https://moodle.gbssg.ch/mod/page/view.php?id=871) up/down mit den Schaltern -d und --build korrekt anwenden

Image

Wenn das Basisimage in Dockerhub verfügbar ist, kann es direkt verwendet werden:

|  |
| --- |
| services:  my-service:  image: ubuntu:latest  ... |

build

wird ein eigenes Imag benötigt, kann dieses aus dem angegebenen Dockerfile erstellt werden:

|  |
| --- |
| services:  my-custom-app:  build: /path/to/dockerfile/  ... |

container\_name

Der Name für den resultierenden Container bei docker ps (analog --name bei docker run)

|  |
| --- |
| services:  my-custom-app:  container\_name: my-container  ... |

restart

Dieser Eintrag definiert die Restart-Policy für einen Container. Folgende Werte sind möglich

* **no**: Der Standard, Container wird nicht automatisch gestartet
* **on-failure**: Der Container wird bei einem Fehler (Exit-Code ungleich 0) automatich neu gestartet
* **always**: Der Container wird immer neu gestartet, insbesondere bei einem Reboot. Wenn der Container manuell gestopped wird, startet er neu, wenn der Dockerdaemon neu gestartet wird.
* **unless-stopped**: Ähnlich wie always, wird der Container manuell gestoppt, startet er nach einem Neustart des Dockerdaemons oder einem Reboot nicht mehr neu.

|  |
| --- |
| * services: * my-custom-app: * restart: always * ... |

ports

Um einen Service vom Host aus zu erreichen, wird die Angabe der Portweiterleitungen benötigt:

|  |
| --- |
| services:  my-custom-app:  image: myapp:latest  ports:  - "8080:3000"  - "8081:4000"  ... |

expose

Angabe der Ports, welche die Container für die Kommunikation untereinander benötigen. Wenn ein Basisimage bereits einen Port exponiert, ist diese Angabe nicht nötig.

|  |
| --- |
| services:  network-example-service:  image: karthequian/helloworld:latest  expose:  - "80" |

networks

Hier kann ein unter dem Top-Level-Schlüsselwort definiertes Netzwerk referenziert werden.

|  |
| --- |
| services:  network-example-service:  image: karthequian/helloworld:latest  networks:  - my\_network  ...  another-service-in-the-same-network:  image: alpine:latest  networks:  - my\_network  ...  networks:  my\_network:  ipam:  config:  - subnet: 172.17.0.0/24  gateway: 172.17.0.1 |

In der Regel werden bei der Definition des Netzwerkes keine weiteren Optionen angegeben. Docker kümmert sich dann selber um die konkrete Definition:

|  |
| --- |
| services:  network-example-service:  image: karthequian/helloworld:latest  networks:  - my\_network  ...  networks:  my\_network: |

volumes

Gibt an wie Containervolumes auf den Host gemountet werden. Benannte Volumes müssen im Top-Level-Schlüsselwort volumes angegeben werden. Somit können benannte Volumes von mehreren Containern gleichzeitig verwendet werden. Der Zusatz :ro mountet ein Verzeichnis read-only.

|  |
| --- |
| services:  volumes-example-service:  image: alpine:latest  volumes:  - my-named-global-volume:/my-volumes/named-global-volume  - /tmp:/my-volumes/host-volume  - /home:/my-volumes/readonly-host-volume:ro  ...  another-volumes-example-service:  image: alpine:latest  volumes:  - my-named-global-volume:/another-path/the-same-named-global-volume  ...  volumes:  my-named-global-volume: |

Unbenannte Volumes und Volumes in eigene Verzeichnisse müssen nicht im Top-Level volume aufgeführt werden.  
Es können auch einzelen Dateien gemountet werden.

depends\_on

Um Abhängigkeiten zu definieren, kann depends\_on verwendet werden. Die betreffenden Container werden dann zuerst geladen:

|  |
| --- |
| services:  kafka:  image: wurstmeister/kafka:2.11-0.11.0.3  depends\_on:  - zookeeper  ...  zookeeper:  image: wurstmeister/zookeeper  ... |

environment

Variablen können sowohl statisch als auch dynamisch mit ${} definiert werden

|  |
| --- |
| services:  database:  image: "postgres:${POSTGRES\_VERSION}"  environment:  DB: mydb  USER: "${USER}" |

Die dynamischen Variablen können dabei u.a. in einer Datei .env im selben Verzeihnis als Key-Value-Paare definiert werden:

|  |
| --- |
| POSTGRES\_VERSION=alpine  USER=foo |

Die verfügbaren Umgebungsvariablen müssen in der Dokumentation zu einem Image auf Dockerhub nachgeschaut werden

secrets

Sollen Passwörter aus sicherheitstechnischen Gründen nicht in der docker-compose Datei aufgeführt werden, kann man secrets verwenden. Das Top-Level-Schlüsselwort gibt an, wo sich die Passwortdatei befindet. Die Einträge bei den Services referenzieren dann die Top-Level-Definition

|  |
| --- |
| services:  db:  image: mysql:latest  environment:  MYSQL\_ROOT\_PASSWORD\_FILE: /run/secrets/db\_root\_password  MYSQL\_PASSWORD\_FILE: /run/secrets/db\_password  MYSQL\_DATABASE: wordpress  MYSQL\_USER: wordpress  secrets:  - db\_root\_password  - db\_password  ...  secrets:  db\_password:  file: db\_password.txt  db\_root\_password:  file: db\_root\_password.txt |

Die in MYSQL\_ROOT\_PASSWORD\_FILE und MYSQL\_PASSWORD\_FILE angegebenen Dateien innerhalb des Containers enthalten dann die in den unter dem Top-Level-Schlüsselwort secrets angegebenen Passwörter. Die verfügbaren Secretsvariablen müssen in der Dokumentation zu einem Image auf Dockerhub nachgeschaut werden.

# Sie können für Sie unbekannte Images anhand verfügbarer Dokumentationen korrekte [docker compose](https://moodle.gbssg.ch/mod/page/view.php?id=871) Kompositionen aufbauen

# Sie können Dockerfiles für Multistage Builds im Zusammenhang mit dotnet aufbauen

# Sie können Dockerfiles im Zusammenspiel mit [docker compose](https://moodle.gbssg.ch/mod/page/view.php?id=871) korrekt verwenden

# Sie können mit dotnet Web- und Konsolen-Anwendungen korrekt umgehen und containerisieren