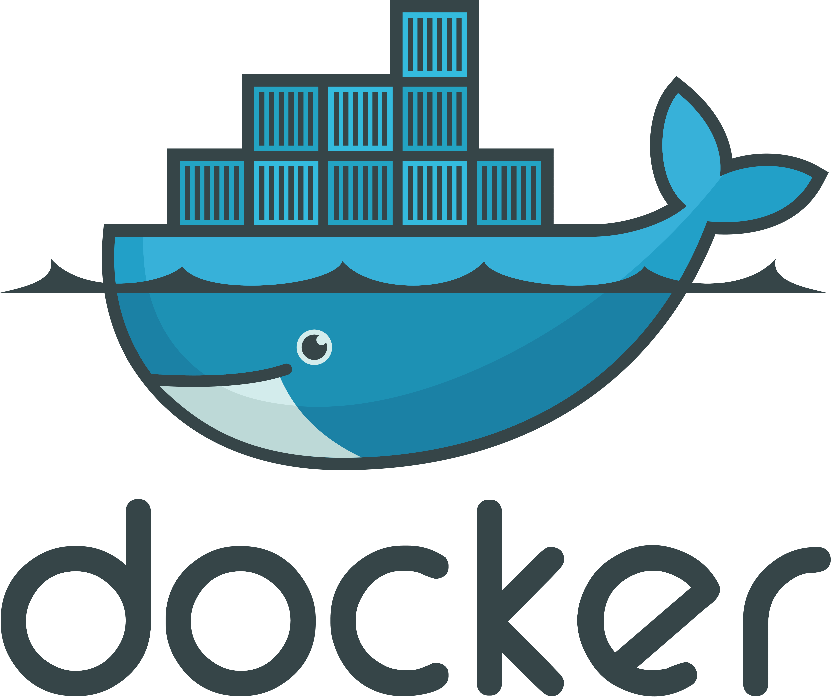
Dokumentation vom Modul 169 Tag 3

Erstellt von: Vladan Vranjes

Klasse Inf2022c

Kontakt: [vvr140992@stud.gibb.ch](mailto:vvr140992@stud.gibb.ch)



Contents

[Kurze Beschreibung was Docker Compose ist 1](#_Toc311883960)

[Version 2 der App mit Docker Compose zum laufen gebracht. 2](#_Toc1947251421)

[Vorgehen, Befehle und Compose Datei im Portfolio festgehalten" 2](#_Toc271353067)

[Portainer installiert und mit PrintScreen festgehalten 3](#_Toc275255703)

[App via Portainer installiert und im Portfolio vorgehen dokumentiert. 4](#_Toc261214180)

["Das Learning Beispiel Shop via Docker Compose installiert. 5](#_Toc1621203897)

[Shop Vorgehen im Portfolio festgehalten. 5](#_Toc1077698302)

[Web-Shop erstellen 5](#_Toc475964128)

[Web-Shop verwalten (BlazorClient) 6](#_Toc1033523265)

[Datenbank (Mongo / Mongo Express) 7](#_Toc102717483)

[Logs (Seq) 8](#_Toc1333666630)

[Message-Broker (RabbitMQ) 8](#_Toc1110979833)

[Analyse einer Transaktion (Jaeger UI) 9](#_Toc1642620840)

[Service Status & Graphische Darstellung eines Verkaufs (Prometheus) 10](#_Toc186246440)

[Dashboard (Grafana) 11](#_Toc1186949323)

# Kurze Beschreibung was Docker Compose ist

Docker Compose ist ein Tool, das es ermöglicht, mehrere Docker-Container mit einer einzigen Konfigurationsdatei zu definieren und zu verwalten. Es erleichtert die Orchestrierung von Containern, die zusammenarbeiten müssen, indem es ihre Konfiguration zentralisiert und das Starten, Stoppen und Skalieren der Container vereinfacht.

# Version 2 der App mit Docker Compose zum laufen gebracht.

Ein Bild, das Text, Screenshot, Zahl, Software enthält.

Automatisch generierte Beschreibung

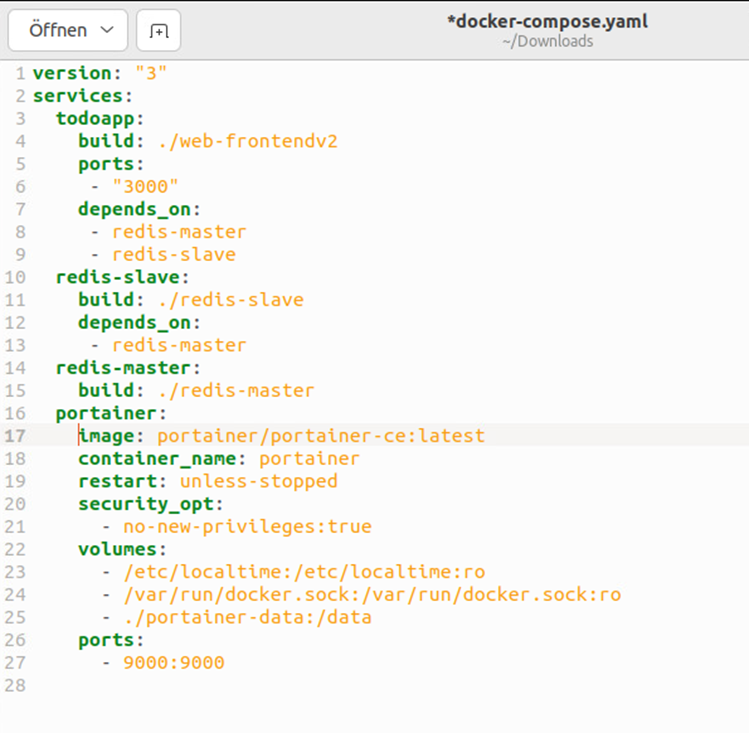
# Vorgehen, Befehle und Compose Datei im Portfolio festgehalten"

1. Docker Compose installation:
   1. sudo apt install docker-compose
2. Wechsle in das Verzeichnis mit dem Docker Compose Yaml File
3. **Passe web-frontend -> web-frontendv2 an**
4. **Speichern**
5. **Ausführen**
   1. **docker-compose -f docker-compose.yaml up -d**

Ein Bild, das Text, Screenshot, Schrift, Software enthält.

Automatisch generierte Beschreibung

# Portainer installiert und mit PrintScreen festgehalten

1. Wechsle in das Verzeichnis mit dem Docker Compose Yaml File
2. **Füge wie im Folgenden Bild den Portainer Block hinzu**
   1. ****
3. **Speichern**
4. **Ausführen**
   1. **docker-compose -f docker-compose.yaml up -d**
5. Ein Bild, das Text, Screenshot, Schrift enthält.

   Automatisch generierte Beschreibung
6. Ein Bild, das Text, Screenshot, Software, Zahl enthält.

   Automatisch generierte Beschreibung
7. Suche auf Localhost :9000
   1. Ein Bild, das Text, Screenshot, Software, Webseite enthält.

      Automatisch generierte Beschreibung
8. Erstelle einen User Welcome#1000
   1. Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

      Automatisch generierte Beschreibung

# App via Portainer installiert und im Portfolio vorgehen dokumentiert.

# A screenshot of a computer Description automatically generated

Danach siehst du es unter Stacks: portainer-v1-vladan

A screenshot of a computer

Description automatically generated

# "Das Learning Beispiel Shop via Docker Compose installiert.

# Shop Vorgehen im Portfolio festgehalten.

## Web-Shop erstellen

* Eintrag in Host-Datei

sudo nano /etc/hosts

* Eintrag: 127.0.0.1 host.docker.internal

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

* Repo herunterladen

git clone <https://git.gibb.ch/thomas.staub/microservices>

* Docker Compose Datei starten

cd microservices/Play.Infra/docker

docker-compose -f docker-compose.yaml up -d

Ein Bild, das Text, Screenshot, Software, Zahl enthält.

Automatisch generierte Beschreibung

* Google: <http://host.docker.internal:5008/>
* Login erstellen

## Web-Shop verwalten (BlazorClient)

* Produkte zum Katalog hinzufügen
  + Catalog ->ADD -> Name, Description, Price

A

A screenshot of a computer

Description automatically generated

* Guthaben für User
  + Users -> Edit -> Value (Gil)

Ein Bild, das Text, Software, Screenshot enthält.

Automatisch generierte Beschreibung

## Datenbank (Mongo / Mongo Express)

* Damit die Daten wie z.B. unsere Einkäufe gespeichert sind, müssen sie ausserhalb des Containers gespeichert werden. Dazu verwenden wir die Mongo DB
* Google: http://host.docker.internal:8080/
* Hier haben wir eine Übersicht über unsere Datenbanken

Ein Bild, das Text, Screenshot, Software, Zahl enthält.

Automatisch generierte Beschreibung

Unter Database: “Catalog -> Collection: items” können wir unsere Artikel sehen welche wir in unserem Shop erfasst habe:

A screenshot of a computer

Description automatically generated

## Logs (Seq)

* Google: http://host.docker.internal:8085/
* Mit Seq können wir die Logs verschiedener Services zentral an einem Ort sehen

Message-Broker (RabbitMQ)

## Message-Broker (RabbitMQ)

* RabbitMQ ist eine Open-Source-Message-Broker-Software, die die Nachrichtenkommunikation zwischen verschiedenen Anwendungen oder Diensten ermöglicht.
* Ein Message Broker ist eine Middleware, die Nachrichten zwischen Anwendungen oder Diensten vermittelt, indem sie Nachrichten empfängt, speichert und weiterleitet.
* Bei Microservices benötigt man einen Message Broker, um die lose Kopplung zwischen den einzelnen Diensten zu ermöglichen und die Kommunikation zu vereinfachen.
* Ein Message Broker ist bei der Skalierung von Microservices wichtig, da er eine verteilte und zuverlässige Kommunikation zwischen den Diensten ermöglicht und Engpässe in der Kommunikation vermeidet.
* Google: host.docker.internal:15672
* Unter Message rates sehen wir wann eine Bestellung getätigt wurde.

Ein Bild, das Text, Software, Zahl, Schrift enthält.

Automatisch generierte Beschreibung

## Analyse einer Transaktion (Jaeger UI)

* Google: http://host.docker.internal:16686/
* Mit «Jaeger» können wir unsere Transaktionen genauer analysieren und sehen jeweils welche Services gebraucht wurden, wie lange die Transaktion verlief, wann diese stattgefunden hat usw.
* Dazu können wir uns unser Log Service (Seq) zur Hilfe nehmen. Dort suchen wir nach dem Log «Received purchase request of (x) item». Dieses Log beinhaltet die Informationen für unsere Transaktion

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

Automatisch generierte Beschreibung

* Nun können wir im Suchfeld oben rechts die TraceID eingeben und unsere Transaktion untersuchen

Ein Bild, das Text, Software, Webseite, Screenshot enthält.

Automatisch generierte Beschreibung

## Service Status & Graphische Darstellung eines Verkaufs (Prometheus)

* Google: http://host.docker.internal:9090/
* Unter Status -> Targets können wir den Status unserer Services sehen

A screenshot of a computer

Description automatically generated

* Unter Graph -> Weltkugel -> Purchase Started können wir unsere Verkäufe graphisch dargestellt sehen.

A screenshot of a graph

Description automatically generated

## Dashboard (Grafana)

Mit Grafana können wir uns eine bessere Übersicht über unsere Services schaffen. Mit so genannten Dashboards können wir dann individuell unsere Graphische Ansicht gestalten. Als Hilfe holen wir uns den Service «Prometheus» hinzu. Diesen kennen wir bereits als wir unsere Verkäufe in einem Graph dargestellt haben. Dazu erstellen wir eine Verbindung zwischen Grafana und Prometheus.

* Google: http://host.docker.internal:4000/

A screenshot of a computer

Description automatically generated

* Dashboard erstellen:

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung