> GitLab now enforces expiry dates on tokens that originally had no set expiration date. Those tokens were given an expiration date of one year later. Please review your personal access tokens, project access tokens and group access tokens to ensure you are aware of upcoming expirations. Administrators of GitLab can find more information on how to identify and mitigate interruption, please take a look at our documentation.

Last edited by **Dominik Arnolds** 4 months ago

Docker

Installation:

- Installiere Docker Desktop von der offiziellen Seite: https://www.docker.com/products/docker-desktop/
- Erstelle dir gegebenenfalls einen kostenlosen Docker-Account

Getting Started:

Vorbereitung:

- Klone das neue Repository, welches alle benötigten Docker-Files enthält
- Wechsele in das Projektverzeichnis
- Dupliziere `.env-example`
- Benenne das Duplikat in `.env` um

> ⚠️ Wichtig: Setzen der Variablen aus `.env`

- Setze den Inhalt von DB_ROOT_PASS auf denselben Wert wie den von DB_PASS

Container starten:

- Starte Docker Desktop App
- `Konsolenbefehl:` docker compose up -d
  -> Eure Anwendung ist nun erreichbar. Z.B. über http://localhost:5000/

Container löschen:

- `Konsolenbefehl:` docker compose down

Container neustarten (zum Testen von geändertem Code):

- `Konsolenbefehl:` docker compose down
- `Konsolenbefehl:` docker compose up -d

Datenbank löschen bzw. zurücksetzen:

- Lösche das Volume "wahlprojekt-deutschlandstipendium_database-data"
  -> `Konsolenbefehl:` docker volume rm wahlprojekt-deutschlandstipendium_database-data
- Lösche nun den Container und starte ihn erneut (siehe oben)
- db-shema.sql wird nun automatisch bei Start des Containers ausgeführt

> Hinweis: Bei erstmaligem Containerstart werden alle Skripte im Verzeichnis /db-init in alphabetischer Reihenfolge ausgeführt

Weitere nützliche Tools:

Images anzeigen:

- `Konsolenbefehl:` docker images

Container anzeigen:

- `Konsolenbefehl:` docker ps

-> alle Container

Volumes anzeigen:

- `Konsolenbefehl:` docker volume ls

Ungenutzte Images/Volumes löschen:

- `Konsolenbefehl:` docker image prune
- `Konsolenbefehl:` docker volume prune

In der Docker-Instanz auf mariadb-Konsole einloggen und SQL-Skript ausführen:

- `Konsolenbefehl:` docker exec -it destip-database mariadb -u root -proot
- `Konsolenbefehl:` source ./docker-entrypoint-initdb.d/1_schema.sql
- `Konsolenbefehl:` source ./docker-entrypoint-initdb.d/2_cat_crit.sql
- `Konsolenbefehl:` source ./docker-entrypoint-initdb.d/testdata.sql

Tabellen der Datenbank anzeigen:

- `Konsolenbefehl:` docker exec -it destip-database mariadb -u root -proot
- `MariaDB-Konsolenbefehl:` use wp_destip_db;
- `MariaDB-Konsolenbefehl:` show tables;

In der Docker-Instanz auf Konsole der Anwendung (bash) zugreifen:

- `Konsolenbefehl:` docker exec -it destip-app bash
- `Konsole verlassen:` exit

Image build erzwingen (bei Änderungen an Dockerfile oder package.json):

- `Konsolenbefehl:` docker-compose down -v
- `Konsolenbefehl:` docker-compose up -d --build

Logs der laufenden Container:

- `Konsolenbefehl:` docker compose logs

> **Die Docker Desktop App bietet ebenfalls einen guten Überblick und viele der aufgelisteten Funktionalitäten.**

Erklärung zu Docker:

Docker ist Grundsätzlich in 3 Schichten aufgeteilt. Das Verstehen und Zusammenarbeiten dieser Schichten bildet die Grundlage zur Softwareentwicklung mit Docker.

Container:

- Eigenständige, ausführbare Softwareeinheit, die Code, Laufzeit, Systemwerkzeuge, Bibliotheken und Einstellungen enthält
- Container sind isoliert voneinander
- Bieten eine konsistente Umgebung, unabhängig von der Umgebung, in der sie ausgeführt werden

Images:

- Vorlage, die als Bauplan für Container dient
- Enthält das Betriebssystem, die Laufzeit, Anwendungen und deren Abhängigkeiten
- Können leicht zwischen verschiedenen Umgebungen geteilt und reproduziert werden

Volumes:

- Mechanismen zum Speichern und Verwalten von Daten, die persistiert werden müssen (auch bei Beendigung oder Löschung des Containers)
- Ermöglichen es, Daten zwischen Containern zu teilen oder Daten außerhalb des Containers zu speichern
  -> Bieten somit eine Möglichkeit, Daten über den Lebenszyklus von Containern hinweg zu erhalten

Docker-Files:

Dockerfile:

- Enthält Anweisungen um ein Docker-Image zu erstellen
- Definiert die Umgebung, in der die Anwendung ausgeführt werden soll und enthält Schritte zum Installieren von Abhängigkeiten, Kopieren von Dateien, usw.

docker-compose.yml:

- Tool zum Definieren und Ausführen von Multi-Container Docker-Anwendungen
- Ermöglicht es, mehrere Container, deren Netzwerke, Volumes und andere Dienste zu definieren

Weitere Einstiegsmöglichkeiten: https://docs.docker.com/get-started/

Konfiguration des Wahlprojektes:

Das "wahlprojekt-deutschlandstipendium" ist so konfiguriert, dass lokale Änderungen an der Anwendung zur Laufzeit des Containers in die entsprechende Docker-Instanz synchronisiert werden. Zum Testen des geänderten Codes muss der Container neugestartet werden (siehe oben). Desweiteren ist es möglich, ohne Seiteneffekte "npm install" im Repository auszuführen, um die Anwendung lokal zu betreiben (DB_HOST Variable anpassen; siehe oben). Es ist aber nicht notwendig, dass Node-Express oder MariaDB lokal installiert ist, um die Anwendung im Docker-Container auszuführen. Die Datenbank in Docker ist durch das definierte Volume persistent. Das SQL-Skript wird nach erstmaligem Containerstart automatisch ausgeführt. Wenn Änderungen in der Dockerfile auftreten oder das Image mit dem "--build"-tag neu aufgebaut wird, können sich einige ungenutzte Images oder Volumes ansammeln. Diese können unbedenklich gelöscht werden (siehe oben).

> ⚠️ Vorsicht bei Änderungen an der package.json. Hier muss der Container gelöscht und das Erstellen des Images erzwungen werden (siehe oben).

Wie funktioniert der Image-Aufbau in Docker?

Da ein Image die Grundlage für einen Container darstellt, muss dieses vorhanden sein, bevor der Container starten kann. Wenn Docker ein Image erstellt, geht es Schritt für Schritt die Dockerfile durch. Dabei hashed und cashed Docker jeden dieser Schritte. Bei Änderungen des Image-Bauplanes muss somit nicht das komplette Image neu aufgebaut werden, sondern es werden nur die Schritte, ab dem Schritt der Änderung, ausgeführt. Dies macht Docker besonders effizient. Wird zum Beispiel nur Code an eurer Server-App geändert, müssen dafür nicht das offizielle Node-Image oder die Dependencies erneut installiert werden. Docker erhält diese Informationen dann aus dem Cache.

Änderungen am Repository:

- db-shema.sql existiert nicht mehr
- Das Schema wurde von den Testdaten und den vorgefertigten Daten zu Kriterium und Kategorie getrennt
- Alle relevanten SQL-Dateien liegen nun im Verzeichnis ./db-init
- .env-mariadb-example, Dockerfile, docker-compose.yml, .dockerignore wurden hinzugefügt