Technische Berufsschule Zürich TBZ

KN01: Docker Grundlagen

Beachten Sie die allgemeinen Informationen zu den Abgaben.

Grundlagen für diesen Auftrag:

- TBZ: Arbeitsumgebung
- TBZ: Docker CLI
- Docker: Referenz CLI
- TBZ: Registry

A) Installation (20%)

Bevor Sie beginnen, stellen Sie sicher, dass Sie verstanden haben was Sie anschliessend abgeben müssen.

Installieren Sie die Umgebung. Eine Anleitung für die notwendigen Tools finden Sie in den Gitlab-Unterlagen.

Abgaben

- Screenshot der Webseite, nachdem Sie den ersten Container erstellt haben.
- Screenshot der Containers in Docker Desktop, der Ihren Container zeigt.

B) Docker Command Line Interface (CLI) (50%)

Sie lernen hier den Umgang mit dem Docker CLI und machen so die ersten Schritte. Als Grundlage dient ihnen das Gitlab-Dokument über die gängigsten Befehle. Führen Sie folgende Schritte aus. **Achtung**: Sie müssen zwischendurch Screenshots erstellen, bevor Sie die Container stoppen und löschen. Schauen Sie auch zuerst nach, welche Abgaben notwendig sind.

1. Überprüfen Sie die Docker-Version. Welchen Befehl müssen Sie dafür verwenden?
2. Suchen Sie nach dem offiziellen **ubuntu** und **nginx** Docker-Image auf Docker Hub mit dem Befehl `docker search`.
3. In Teil A mussten Sie den Befehl `docker run -d -p 80:80 docker/getting-started` ausführen. Erklären Sie die verschiedenen Parameter.

4. Mit dem **nginx** Image verfahren Sie wie folgt. Wir zeigen, dass der Befehl `docker run`, das gleiche ist wie die drei Befehle `docker pull`, `docker create` und `docker start` hintereinander ausgeführt.
    1. Laden Sie das Image herunter mit dem Befehle `docker pull`
    2. Erstellen Sie ein Container mit dem Befehl `docker create`. exponieren Sie den Port 8081 mit dem Mapping auf den Port 80, so dass Sie anschliessend die URL http://localhost:8081 aufrufen können.
    3. Starten Sie das Image mit dem Befehl `docker start`. Evtl. müssen Sie einen laufenden Container zuerst stoppen, weil der Port bereits verwendet wird.
    4. Erstellen Sie einen **Screenshot** der Standard-Seite von nginx mit der URL sichtbar.
5. Mit dem **ubuntu** Image verfahren Sie wie folgt. Wir zeigen, dass nicht jedes Image im Hintergrund ausgeführt werden kann.
    1. Erstellen und starten Sie einen Container mit dem Befehl `docker run -d`. **Kommentieren** Sie was mit dem Container geschieht in 3-5 Sätzen. wurde das Image automatisch heruntergeladen? Konnte es starten?
    2. Erstellen und starten Sie einen Container mit dem Befehl `docker run -it`. **Kommentieren** Sie was nun geschieht in 3-5 Sätzen.
6. Stellen Sie sicher, dass Ihr **nginx**-Container **bereits läuft**. Öffnen Sie nun nachträglich eine interaktive Shell. Der Unterschied zu vorher ist, dass Sie nicht den Container mit interactiver Shell starten, sondern eine Shell eines laufenden Containers öffnen. Der Befehl ist `docker exec -it <name-ihres-container> /bin/bash`
    1. Führen Sie den Befehl `service nginx status` aus. Erstellen sie einen **Screenshot** des Befehls und des Resultats. Sie sehen, dass Sie sich innerhalb des Docker-Images bewegen können.
    2. Beenden Sie die interaktive Shell im Docker-Container (mit dem Befehl `exit`). Beachten Sie, dass das Image automatisch wieder beendet.
7. Überprüfen Sie den Status der Container. Erstellen Sie einen **Screenshot** des Befehls und des Resultats.
8. Stoppen Sie nun noch den Container des nginx Images mit dem entsprechenden Docker-Befehl
9. Entfernen Sie alle Container mit dem entsprechenden Docker-Befehl.
10. Entfernen Sie die beiden Images von Ihrer lokalen Umgebung mit dem entsprechenden Docker-Befehl

Abgaben

- Datei, die alle Befehle auflistet, die Sie eben verwendet haben.
- Erklärungen zu den Befehlen, falls erwartet.
- Screenshots gem. den Bemerkungen in der Ausführungsliste.

C) Registry und Repository (10%)

Alle Images, welche Sie bisher verwendet haben, sind aus öffentlichen Repositories. Sie können aber auch Images in eigenen Repositories führen, sowohl öffentlich als auch privat.

Erstellen Sie einen Account unter Docker Hub mit Ihrer **TBZ-Email**. Erstellen Sie dort das **private** Repository mit dem Namen **m347**.

Beachten Sie die Namensform *benutzername/reponame*. Diesen werden Sie später benötigen.

Logen Sie sich in Docker Desktop ein mit Ihrem neuen Benutzer, damit Sie Zugriff auf das private Repository haben.

Abgaben:

- Screenshot mit dem leeren Inhalt des neu erstellten Repository

D) Privates Repository (20%)

In KN01 hatten Sie mit den öffentlichen Images **nginx** und **ubuntu** gearbeitet. Laden Sie das öffentliche Image für **nginx** nochmals herunter. In Docker Desktop sehen Sie, dass es den Tag **latest** enthält. In den Unterlagen haben Sie gelesen, dass Tags oft für die Versionierung verwendet werden. Sie können aber in einem Repository auch eine Applikation zusammenfassen. Führen Sie folgende Schritte durch:

- Wir "kopieren" das Image für nginx in das eigene Repository mit dem Befehl `docker tag nginx:latest Benutzername/reponame:nginx`. Erklären Sie was dieser Befehl genau macht. Was ist ein Tag?
- Führen Sie den Befehl `docker push Benutzername/reponame:nginx` aus. Erklären Sie was dieser Befehl genau macht.
- Laden Sie das öffentliche Image für mariadb herunter und erstellen Sie wieder den entsprechenden Tage wie im Schritt vorher
- Pushen Sie alles in Ihr Repository hoch mit dem entsprechenden Befehl.

Abgaben:

- Screenshot der Docker-Hub Seite mit den sichtbaren Tags Ihres Repos
- Befehle, die Sie ausgeführt haben
- Erklärung zu den Befehlen, die die einzelnen Komponenten beschreiben.