Inhalt der Checkliste

Inhalt der Checkliste **1**
Projektabgabe Conduit Container **2**
1. Repository 2
.gitignore Datei 2
Dockerfiles 2
.dockerignore Datei 2
docker-compose.yaml 2
README.md 2
2. Dokumentation 3
3. Hinweise 3

Projektabgabe - Conduit-Application

Bitte erfülle alle Punkte auf dieser Liste, bevor du das Projekt einreichst. Solltest du weitere Extras eingebaut haben, erwähne das kurz, damit sich die Mentoren dies bei Bedarf anschauen können.

1. Repository

Vorhandene Dateien

- ☒ Es wurde eine .gitignore Datei angelegt, die alle irrelevanten Inhalte aus dem git repository ignoriert
- ☒ Es gibt jeweils ein Dockerfile für die Inhalte der Backend- bzw. Frontend-Applikation, welches den Anforderungen aus dem nächsten Punkt genügt
- ☒ Es gibt eine docker-compose.yaml, die den Anforderungen des übernächsten Punkts genügt
- ☒ Eine Datei namens README.md ist vorhanden und entsprechend der Kriterien unten erstellt worden

Dockerfiles

- ☒ Die Dockerfiles basieren auf einem passenden Basis-Image für den jeweiligen Technologie-Stack (nodejs, python)
- ☒ Notwendige Environment-Variablen werden innerhalb der Dockerfiles konfiguriert
- ☒ Die Dockerfiles exposen einen Container-Port für das Internet
- ☒ Die Dockerfiles verwenden Multi-Stage Builds im die resultierende Container-Image-Größe gering zu halten

.dockerignore Dateien

- ☒ Es gibt eine .dockerignore Datei, die nach dem selben Prinzip wie die gitignore Datei Inhalte listet, die bei einem Containerbuild ignoriert und nicht in das Image kopiert werden sollen (z.b. node_modules, etc.)

docker-compose.yaml

- ☒ Es gibt je einen Service, der definiert und konfiguriert wird: frontend, backend
- ☒ Es gibt eine Env Konfiguration für die beiden Services, sodass alle unkritischen Variablen konfiguriert werden (keine Auth!)
- ☒ Es gibt notwendige Port-Freigaben und entsprechende Mappings, sodass die Container aus dem Internet erreichbar sind
- ☒ Es gibt Volumes-Konfigurationen für die Container Daten, sodass die Inhalte auf einem Dateisystem persistiert werden und der Datenstand nicht verloren geht

README.md

- ☒ Die README sollte ein Inhaltsverzeichnis a.k.a. eine Table-of-Contents (ToC) enthalten
- ☒ Eine Sektion mit einer Beschreibung des Repositories muss vorhanden sein. In dieser Beschreibung sollte genannt werden was die wesentlichen Inhalte sind, was der Zweck des Repositories ist
- ☒ Eine Sektion "Quickstart" sollte als Teil der README enthalten sein. Hier sollten kurz Voraussetzungen genannt und eine Schnellstart-Anleitung beschrieben sein.
- ☒ Es ausführliche Variante der vorgenannten Sektion so als "Usage" enthalten sein. Hier soll genauer auf die Konfiguration und Konfigurierbarkeit eingegangen werden, d.h. es soll auch erklärt werden wie relevante Passagen modifiziert werden können, um andere Resultate zu erzielen.

2. Dokumentation

Die Dokumentation des Codes, sowie des Projektes soll im Repository in Form einer README Datei stattfinden.
Die Dokumentationssprache für alle Projekte (und zugehörige Unterlagen) ist englisch.

3. Hinweise

Sicherheitshinweise

- ☒ Speichere keine SSH-Keys im Workspace deines Git-Repositories
- ☒ Speicher keine Passwörter, Tokens, oder Benutzernamen in deinem Code. Verwende hierfür stattdessen Environment-Variablen
- ☒ Speicher keine IP-Adressen, oder sonstigen sensiblen Informationen in einem Git Repository

Code-Konventionen

- ☒ Für build-args, environment Variablen und Shell-Variablen gilt folgende Namenskonvention: UPPER_CASE_WITH_UNDERSCORE
- ☒ Bei einer Referenz auf eine Variable sollte immer die {}-Notation verwendet werden um Fehler in der Interpretation zu vermeiden: ${SOME_VAR_VALUE}, statt: $SOME_VAR_VALUE
- ☒ Es sollten für build-args, oder Environment Variablen "Default"-Werte konfiguriert werden, allerdings nur dann wenn dies Sinn ergibt.
- ☒ Kritische Konfiguration wie Tokens, Passwörter oder ähnliches sollte nicht im Code-Repository gespeichert sein, sondern bspw. durch die Verwendung eines .env-files in einen Container hineingegeben werden

Testing

Bevor Du dein Projekt einreichst, solltest du die folgenden Dinge sichergestellt und getestet haben:

- ☒ Das Frontend deiner Conduit-Applikation ist erreichbar unter der IP-Adresse deiner Cloud-VM auf Port 8282
- ☒ Dein ENTRYPOINT startet eine WSGI Applikation, KEINEN dev-server!
- ☒ Die Container der Services werden neugestartet, sobald ein Fehler passiert der zum Terminieren des Containers führt.
- ☒ Du kannst durch die Applikation navigieren und es werden überall Daten geladen
- ☒ Du kannst die Logs deiner Anwendung via CLI einsehen und ggf. auch persistieren, d.h. du kannst dir die Logs in einer Datei zur späteren Verwendung abspeichern.

Logs in Datei abspeichern:

```
docker logs [container-name] > meine-container-logs.txt
```

`[container-name]` muss dann entsprechend mit dem Namen des Containers ersetzt werden.