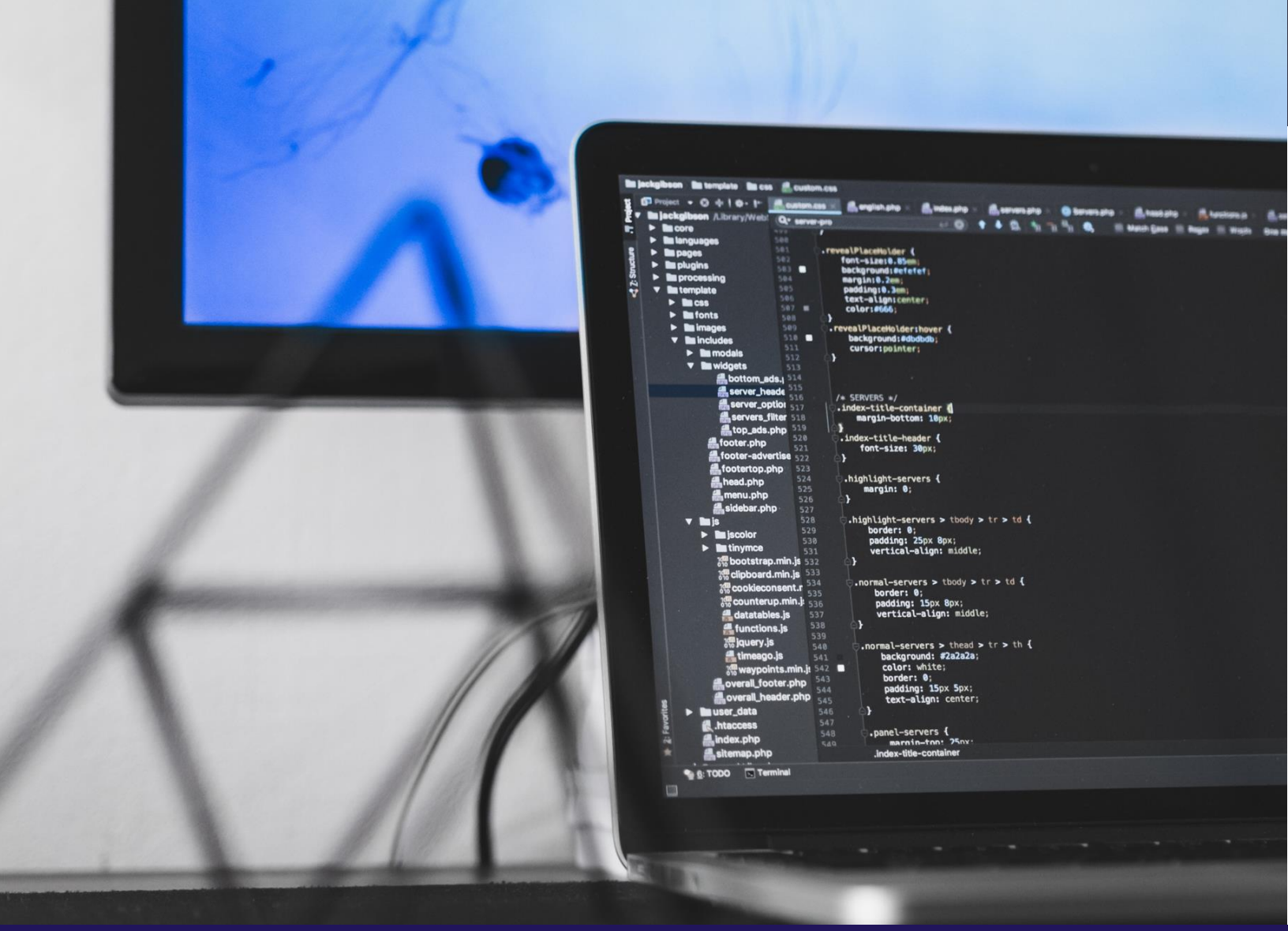
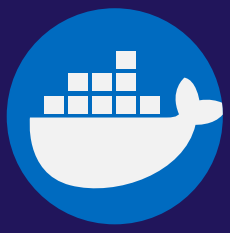


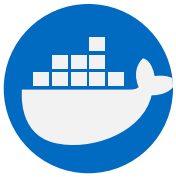
Docker Grundlagen

Kursheft

Kapitel: Abschlussprojekt

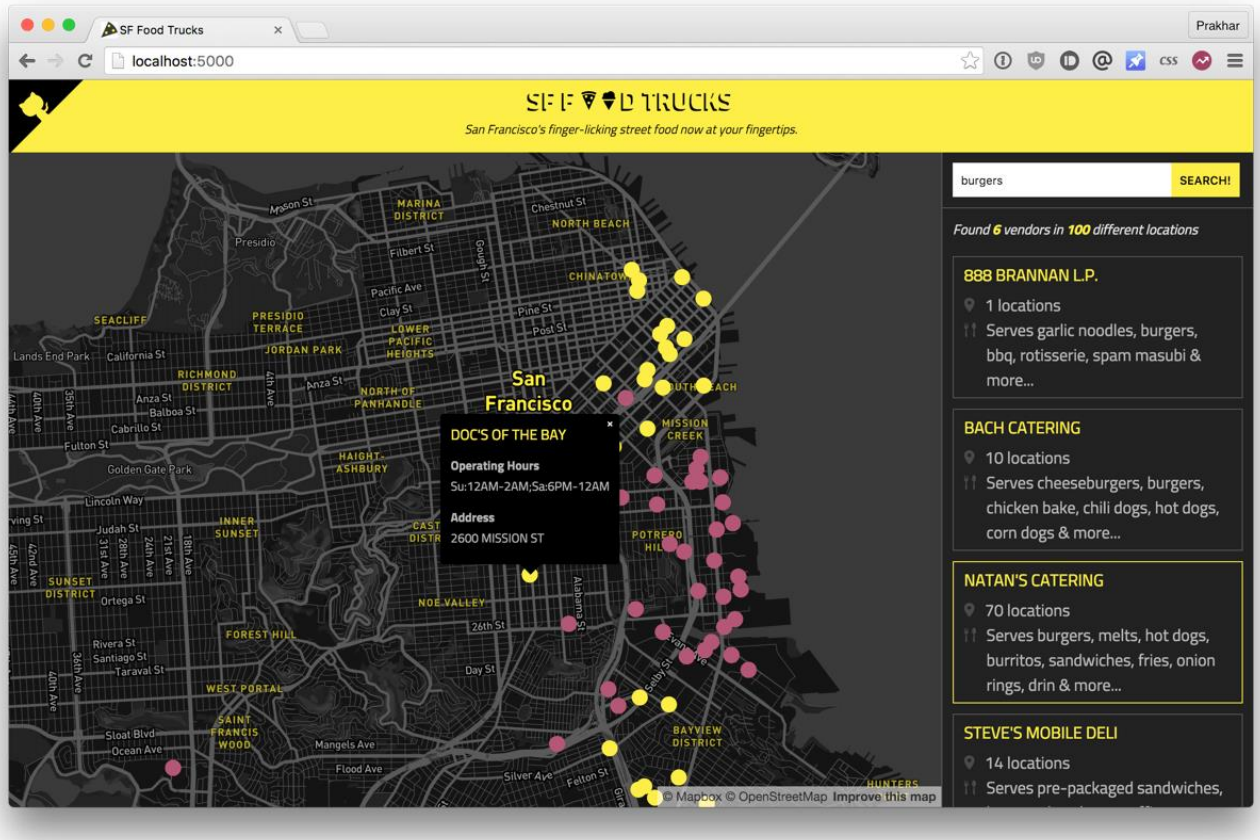


Inhaltsverzeichnis



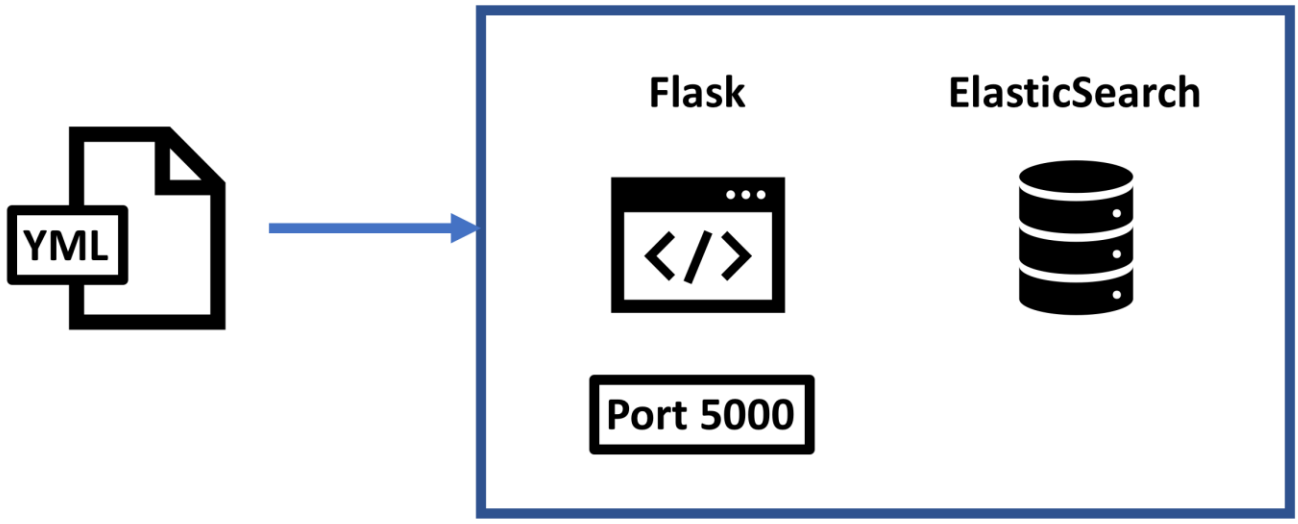
1. Übersicht

Die Webanwendung, die wir jetzt in Docker umsetzten, heißt SF Food Trucks. Mein Ziel beim Erstellen dieser App war es, etwas Nützliches zu haben (da es einer realen Anwendung ähnelt), sich auf mindestens einen Dienst verlässt, aber nicht zu komplex ist:



In diesem Projekt wirst du die Python Webanwendung mit ElasticSearch verbinden. Elasticsearch ist eine Suchmaschine auf Basis von Lucene und speichert Dokumente in einem NoSQL-Format (JSON). Die Kommunikation mit Klienten erfolgt über ein Webinterface.

Wir verwenden dies als unsere Beispielanwendung, um herauszufinden, wie eine Umgebung mit mehreren Containern erstellt, ausgeführt und bereitgestellt wird.



Du kannst sehen, dass die Anwendung aus einem Flask-Backend-Server und einem Elasticsearch-Dienst besteht. Eine natürliche Möglichkeit, diese App zu teilen, wäre, zwei Container zu haben - einen, der den Flask-Prozess ausführt, und einen anderen, der den Elasticsearch (ES)-Prozess ausführt. Der Grund dafür ist, dass wenn unsere App populär wird, wir sie einfach skalieren können. Das wäre ganz einfach, indem wir weitere Container hinzufügen, je nachdem, wo der Engpass liegt. Entweder im Frontend, wenn es viele Nutzer gibt oder im Backend, wenn viele Daten gespeichert werden müssen.

2. Webanwendung

Erstelle jetzt ein leeres Dockerfile:

```
touch Dockerfile
```

Und füge den folgenden Inhalt ein:

```
FROM ubuntu:18.04

LABEL maintainer Mein Name <Mein@Name.me>

# Installiere die systemweiten Abhängigkeiten für Python und nodejs
RUN apt-get -yqq update
RUN apt-get -yqq install python-pip python-dev curl gnupg
RUN curl -sL https://deb.nodesource.com/setup_10.x | bash
RUN apt-get install -yq nodejs

# Kopiere den Code der Anwendung in den Container
COPY flask-app /opt/flask-app
WORKDIR /opt/flask-app

# Installiere die anwendungsspezifischen Pakete
RUN npm install
RUN npm run build
RUN pip install -r requirements.txt

# Lasse den Port anzeigen
EXPOSE 5000

# Starte die Anwendung
CMD [ "python", "./app.py" ]
```

3. Suchmaschine

Hinweis: Da Elastic das Unternehmen hinter Elasticsearch ist, unterhält es ein eigenes Register für Elastic-Produkte: https://www.docker.elastic.co/. Daher wird empfohlen, die Images aus dieser Registry zu verwenden, wenn du Elasticsearch verwenden möchtest.

Lade dir das folgende Image herunter:

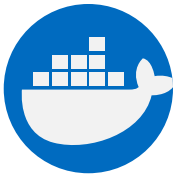
```
docker pull docker.elastic.co/elasticsearch/elasticsearch:6.3.2
```

Erstelle als nächstes die docker-compose Datei:

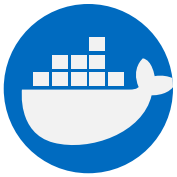
```
touch docker-compose.yml
```

Und füge den folgenden Inhalt hinzu:

```
version: "3"
services:
  es:
    image: docker.elastic.co/elasticsearch/elasticsearch:6.3.2
    container_name: es
    environment:
      - discovery.type=single-node
    ports:
      - 9200:9200
    volumes:
      - esdata1:/usr/share/elasticsearch/data
  web:
    image: .
    depends_on:
      - es
    ports:
      - 5000:5000
```



```
    volumes:
      - ./flask-app:/opt/flask-app
volumes:
    esdata1:
      driver: local
```



4. Starte den Service

Führe jetzt den “Zauberbefehl” aus:

```
docker-compose up -d
```

Lasse dir jetzt die laufenden Container ausgeben:

```
docker ps
```

Öffne deinen Browser und gehe auf den folgenden Link:

- http://localhost:5000

Hinweis: Schaue dir die Videolektion an, um mehr über die Anwendung zu erfahren

Beende jetzt wieder deinen Service

```
docker-compose down -v
```