MICROSERVICE-APPLICATION LAB

Benutzeranleitung

von

Cyrill Jauner, Alan Meile

Hochschule Luzern
Lucerne University of Applied Sciences and Arts

21. Dezember 2018

Inhaltsverzeichnis

1 Zweck dieses Dokuments

In diesem Dokument sind Anleitungen für die Benutzer des Microservices Application Labs aufgeführt. Alle Benutzer sollten in der Lage sein, damit das Lab zu verwalten und auszuführen.

2 Übersicht des Application Labs

Das Application Lab basiert auf einer Microservices-Architektur. Alle Applikationen haben eine gemeinsames Git Repository. Dieses Repository wird auf der Gitlab-Plattform der Hochschule Luzern gehostet. Jeder Service wird als eigener Docker-Container ausgeführt. Dazu stellt jeder Servie eine Dockerdatei zur Verfügung. Darin sind die Informationen zum Builden der Applikation hinterlegt. Von den vorhandenen Containern besitzen drei ein Portmapping. Die Container gateway, client und rabbitmq können dadurch von ausserhalb der VM über den gemappten Port angesprochen werden. Alle übrigen Container haben kein solches Mapping konfiguriert und können daher nur untereinander kommunizieren. Die Abbildung 1 gibt einen Gesamtüberblick des Labs mit der Gitlab-Plattform als Umsystem und möglichen Zugriffen von Benutzern aus dem HSLU-Lan. Die Benutzerbezeichnungen (z.B. Entwickler) sind dabei willkürlich gewählt und sollen lediglich die möglichen Zugriffe illustrieren.

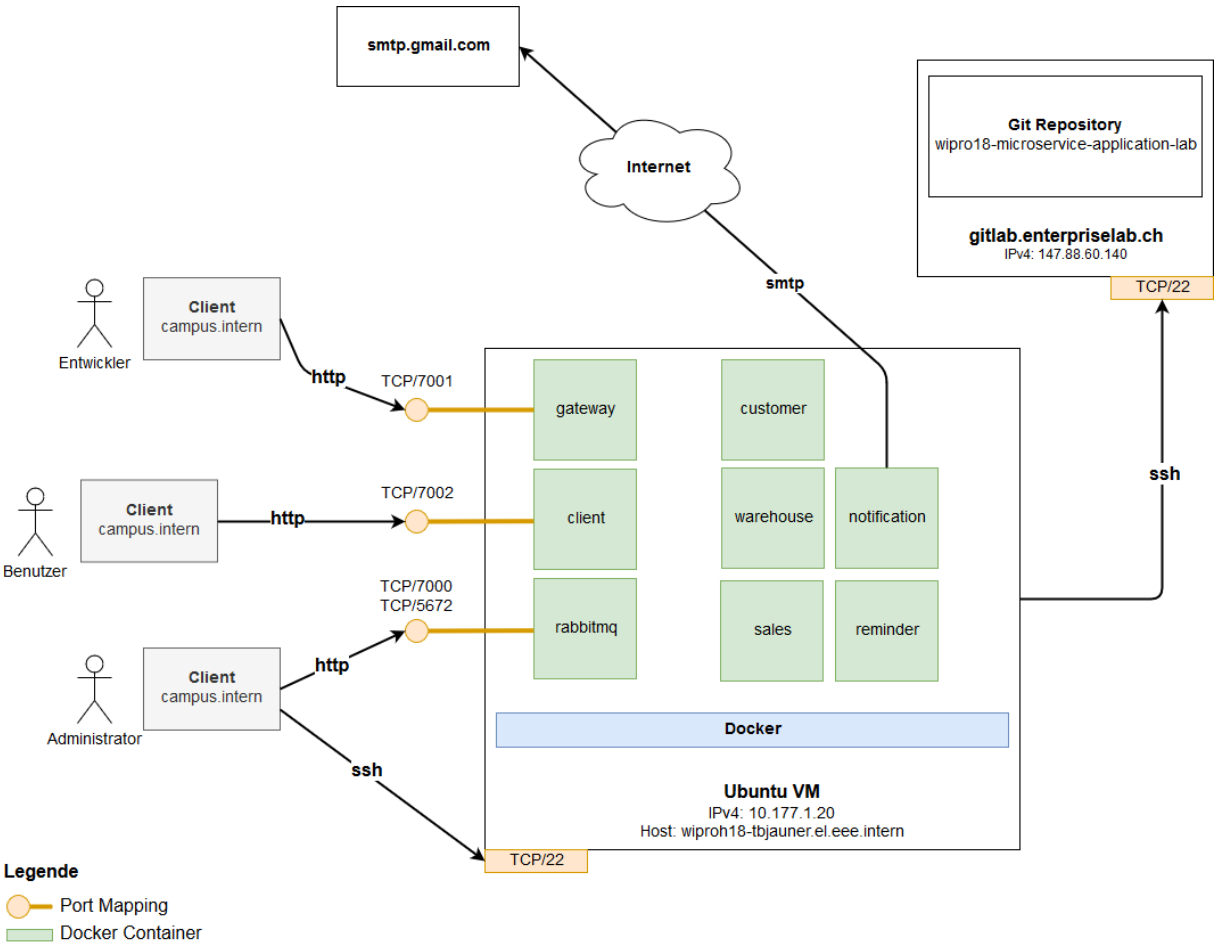

Abbildung 1 Übersicht Application Lab

2.1 Allgemeine Informationen

In diesem Kapitel werden allgemeine Informationen zum Lab aufgeführt.

Virtuelle Maschine

Hostname:	wiproh18-tbjauner.el.eee.intern	
Ipv4 Adresse:	10.177.1.20	
Betriebssystem:	Ubuntu 18.04.1 LTS	
Benutzer (sudo):	Username:	labuser
	Passwort:	labuser
Zugriff:	Nur vom HSLU-Netzwerk aus möglich	

Git Repository

Name:	wipro18-microservice-application-lab
Link auf Gitlab:	https://gitlab.enterpriselab.ch/tbjauner/wipro18-microservice-application-lab
Sichtbarkeit:	Internal: Jeder angemeldete Benutzer kann das Repository lesen und klonen
Deploy-Key:	Konfiguriert für den Zugriff von der VM
	Fingerprint: 46:da:1a:75:f6:e2:3c:c6:95:fd:90:fc:c5:1d:09:7b

RabbitMQ

Version:	3.7.8	
Benutzer:	Username:	guest
	Passwort:	guest

Notification Gmail Account

E-Mail-Adresse:	noreply.wipro18@gmail.com
Passwort:	wipro18-applab

3 Anleitungen

3.1 Zugriff auf die virtuelle Maschine

Diese Anleitung beschreibt den Zugriff auf die virtuelle Maschine. Ausserdem wird die Konfiguration auf der VM gezeigt.

Voraussetzungen

Der Benutzer befindet sich im internen HSLU-WLAN oder ist via VPN mit dem Netzwerk der Hochschule Luzern verbunden. Windows-Benutzer haben auf ihrem Betriebssystem ssh konfiguriert. Falls nicht, sollte derselbe Zugriff auch mit einem ssh-Tool wie z.B. Putty möglich sein.

Vorgehen

1. Terminal (Mac, Linux) oder Powershell (Windows) öffnen.
   Hinweis: Mit dem Update 1803 von Windows 10 wurde OpenSSH installiert und kann via CLI genutzt werden. Weitere Infos: https://blogs.msdn.microsoft.com/commandline/2018/03/07/windows10v1803/

2. In der CLI den Befehl `ssh labuser@wiproh18-tbjauner.el.eee.intern` absetzen. Bei der ersten Verbindung muss dem Zielgerät mit `yes` vertraut werden. Danach das Kennwort `labuser` eintragen. Anschliessend sollte der Benutzer labuser eingeloggt sein.

3. Im Homeverzeichnis des labusers befinden sich zwei relevante Verzeichnisse. Im Verzeichnis `./.ssh` sind die RSA-Schlüssel abgelegt. Der Public Key wurde als Deploy Key im Gitlab-Repository hinterlegt. Das Verzeichnis `./wipro18-microservice-application-lab` ist das Git Repository des Wirtschaftsprojekts. Darin ist das Verzeichnis `application_lab` zu finden.

3.2 Aktualisieren der Applikationen

Diese Anleitung beschreibt wie das Application Lab auf der Virtuellen Maschine aktualisiert werden kann.

Voraussetzungen

Alle Änderungen sind im Git Repository auf dem Gitlab eingepflegt. Der Benutzer ist auf der VM eingeloggt (siehe Anleitung 3.1).

Vorgehen

1. In das Repository Verzeichnis wechseln mit
   ```
   cd /home_static/labuser/wipro18-microservice-application-lab
   ```

2. Das Repository aktualisieren mit dem Befehl `git pull`. Mit dem Befehl `git status` kann verifiziert werden, dass das Repository aktuell ist. Ist dies der Fall wird der folgende Text ausgegeben.
   ```
   On branch master
   Your branch is up to date with 'origin/master'.
   ```

3. Damit sind die Daten auf der VM aktuell. Allerdings werden die Docker Container nicht automatisch aktualisiert. Dazu müssen die Container neu gebuildet und ausgeführt werden (siehe 0).

3.3 Build und Ausführung des Labs (VM)

Diese Anleitung beschreibt wie das Application Lab ausgeführt werden kann. Dabei werden alle Container gestartet.

Voraussetzungen

Der Benutzer ist auf der VM eingeloggt (siehe Anleitung 3.1).

Vorgehen

1. In das Application Lab Verzeichnis wechseln mit
   `cd /home_static/labuser/wipro18-microservice-application-lab/application_lab`

2. In diesem Verzeichnis ist die Datei `docker-compose.yml` abgelegt. Darin werden alle Services definiert, die ausgeführt werden sollen. Zunächst soll sichergestellt werden, dass die Container nicht mehr Laufen. Dazu den Befehl `sudo docker-compose down -v` ausführen. Falls das sudo-Kennwort verlangt wird, das Kennwort `labuser` angeben. Damit sollten alle bestehenden Container gestoppt und entfernt werden.

   ```
   Removing application_lab_notification_1_f9c14f199170 ... done
   Removing application_lab_warehouse_1_a4b1df47c247    ... done
   Removing application_lab_gateway_1_de12ad276323      ... done
   Removing application_lab_customer_1_a8df64742c58     ... done
   Removing application_lab_sales_1_b28c2004db44        ... done
   Removing application_lab_client_1_5d652b87db45       ... done
   Removing application_lab_reminder_1_401878ae374c     ... done
   Removing application_lab_rabbitmq_1_e8830f0cc79a     ... done
   Removing network application_lab_default
   ```

3. Nun kann das Lab mit dem Befehl `sudo docker-compose up --build` gestartet werden. Der Parameter `--build` bewirkt dass die Container neu gebuildet werden anhand der Dockerdatei des Services.
   Hinweis: Fehlende Images auf dem Server müssen zuerst heruntergeladen werden. Daher kann ein vollständiger Build ungefähr sechs Minuten dauern.

4. In der Konsole sollten die Lognachrichten der Services angezeigt werden. Diese sind mit dem Containernamen versehen und farblich markiert.
   ```
   rabbitmq_1_18d0f85a0498 |  * rabbitmq_management_agent
   gateway_1_aafdf9b4f249 | INFO: Started listener bound to [0.0.0.0:8080]
   ```

5. Werden alle Services ausgeführt, kann auf die Website zugegriffen werden. Dazu im Browser die Adresse http://wiproh18-tbjauner.el.eee.intern:7002/ aufrufen. Die Website sollte wie folgt dargestellt werden.

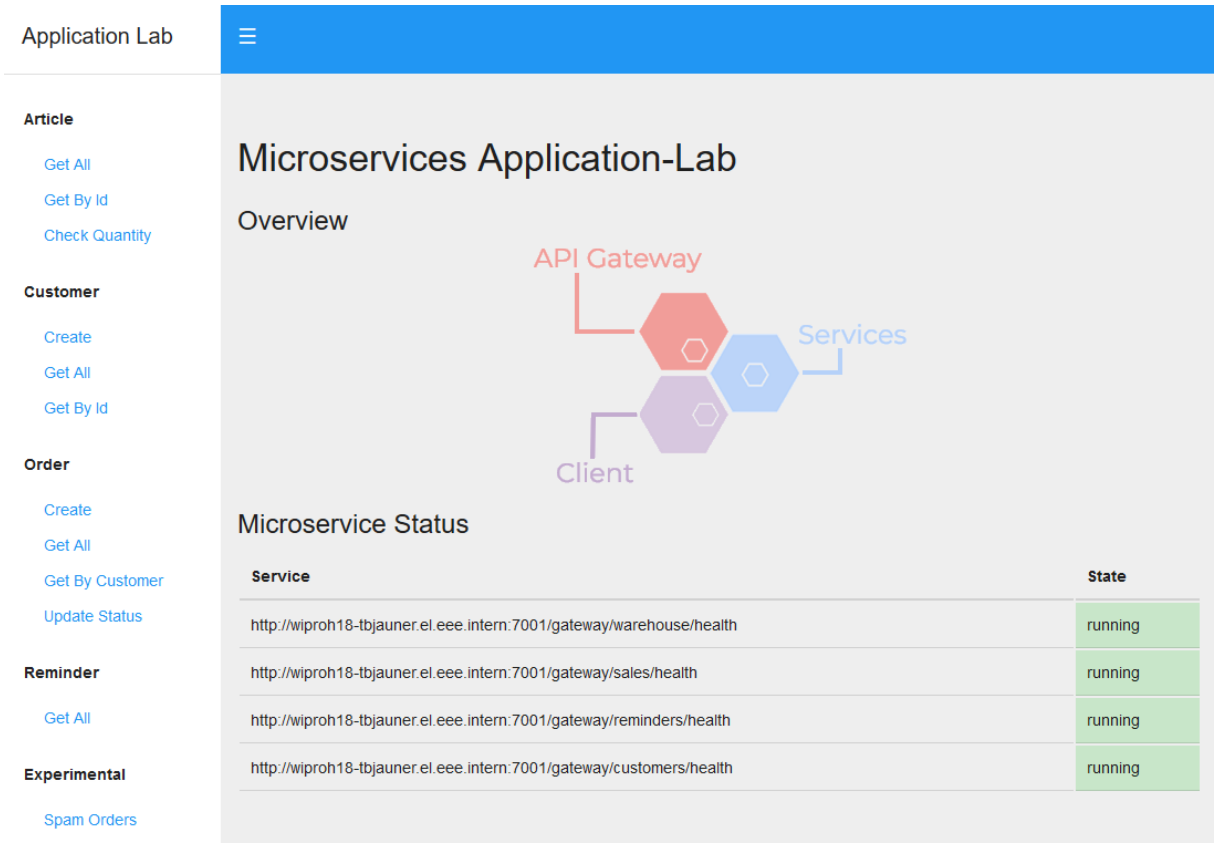

Abbildung 2 Ansicht Webclient

3.4 Build und Ausführung des Labs (Lokal)

Diese Anleitung beschreibt wie das Application Lab auf einem Client gebuildet und ausgeführt werden kann. Ausserdem wird gezeigt, wie ein REST-Client zum Testen verwendet werden kann.

Voraussetzungen

Auf dem Client ist Docker und Docker-Compose installiert. Der Benutzer besitzt ein Login auf der Gitlab-Plattform und hat das Repository lokal geklont. Daneben sollte ein REST-Client installiert sein. Diese Anleitung wurde mit SoapUI durchgeführt.

Vorgehen

1. In das Repository-Verzeichnis `wipro18-microservice-application-lab/application_lab` wechseln.

2. Jetzt können die Punkte zwei bis vier der Anleitung 0 ausgeführt werden. Anschliessend sollte das Lab ausgeführt werden.

3. Nun kann mit dem Lab interagiert werden. Allerdings funktioniert der Webclient lokal nicht, da die Website versucht auf den Ubuntu-Server zu verbinden. Falls die Website Request abschicken kann, landen diese beim Server und nicht bei den lokalen Services.

4. Die Kommunikation mit den Services erfolgt über das API Gateway. Das Gateway stellt eine WADL-Datei zur Verfügung mit allen verfügbaren Schnittstellen. Falls der REST-Client die Methoden über eine WADL-Datei importieren kann, sollte einfach der folgende Link hinterlegt werden http://localhost:7001/gateway/application.wadl

5. Hinweis: Auch, wenn der Client lokal nicht eingesetzt wird, kann mit der Website ein JSON-String zusammengesetzt werden. Dazu muss auf der Website einfach die gewünschte Aktion ausgeführt werden, ohne den Request abzuschicken. Beispielsweise kann so eine Bestellung zusammengeklickt werden. In der Abbildung 3 ist ein Beispiel für einen String einer neuen Bestellung auf der Website aufgeführt. Dieser String kann auch in einem REST-Client verwendet werden.

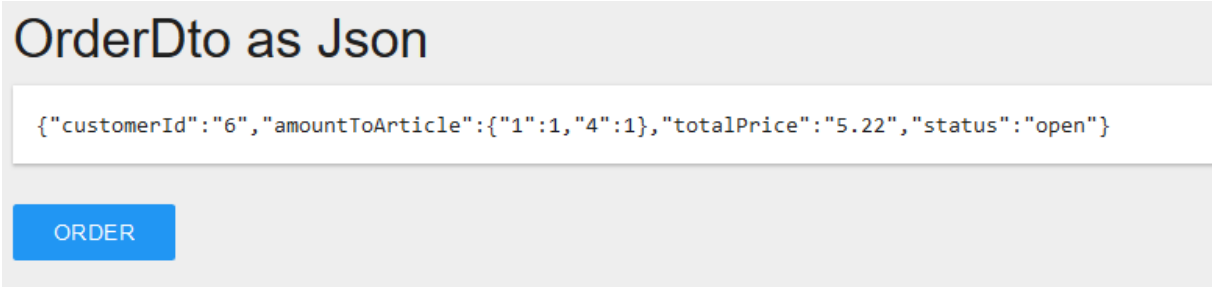

Abbildung 3 JSON String auf der Website

3.5 Firehose Feature für RabbitMQ konfigurieren

Diese Anleitung beschreibt wie das Firehose Feature von RabbitMQ konfiguriert werden kann. Damit lassen sich Messages aufzeichnen, die über den Message Broker verschickt werden.

Voraussetzungen

Das Application Lab wird lokal oder auf der virtuellen Maschine ausgeführt.

Vorgehen

1. Auf dem Computer wo die Container ausgeführt werden muss der CLI Command `sudo docker ps` (Für Windows Shell ohne sudo) ausgeführt werden. Dieser Befehl liefert eine Übersicht aller ausgeführter Container. Aus dieser Übersicht soll die Id des rabbitmq Containers kopiert werden.

2. Nun sollen in der CLI des rabbitmq Containers zwei Befehle ausgeführt werden. Zunächst `sudo docker exec –it <CONTAINER_ID> rabbitmq-plugins enable rabbitmq_tracing`. Dies aktiviert das Trace-Plugin von rabbitmq. Danach ist der Befehl sudo docker exec –it `<CONTAINER_ID> rabbitmqctl trace_on` abzusetzen. Damit sollte nun die Trace-Website freigeschalten sein. Ein erfolgreicher Durchlauf ist auf der Abbildung 4 zu sehen.

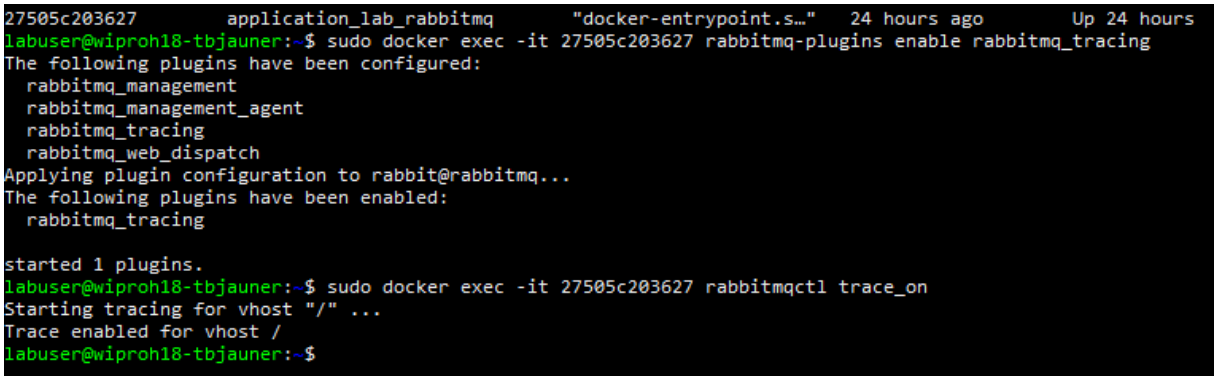

Abbildung 4 Firehose Commands

3. Als nächstes muss eine neue Aufzeichnung eingerichtet werden. Dazu das Web-UI von RabbitMQ öffnen. Auf der virtuellen Maschine http://wiproh18-tbjauner.el.eee.intern:7000 und bei einer lokalen Ausführung http://localhost:7000. Anschliessend kann mit guest / guest eingeloggt werden.

4. In den Abschnitt *Admin* wechseln und rechts das Menu *Tracing* aufrufen. Dies öffnet eine Eingabemaske, um eine Aufzeichnung zu erstellen. Es muss ein Name vergeben werden und ein Benutzer für die Aufzeichnung angegeben werden. Als Benutzer soll guest / guest angegeben werden. Im Feld *Pattern* können die Nachrichten gefiltert werden. Wird das Feld leer gelassen, werden alle Nachrichten aufgezeichnet. Eine mögliche Konfiguration ist in der Abbildung 5 angegeben. Laufende Aufzeichnungen werden oberhalb der Maske unter *All traces* angezeigt.

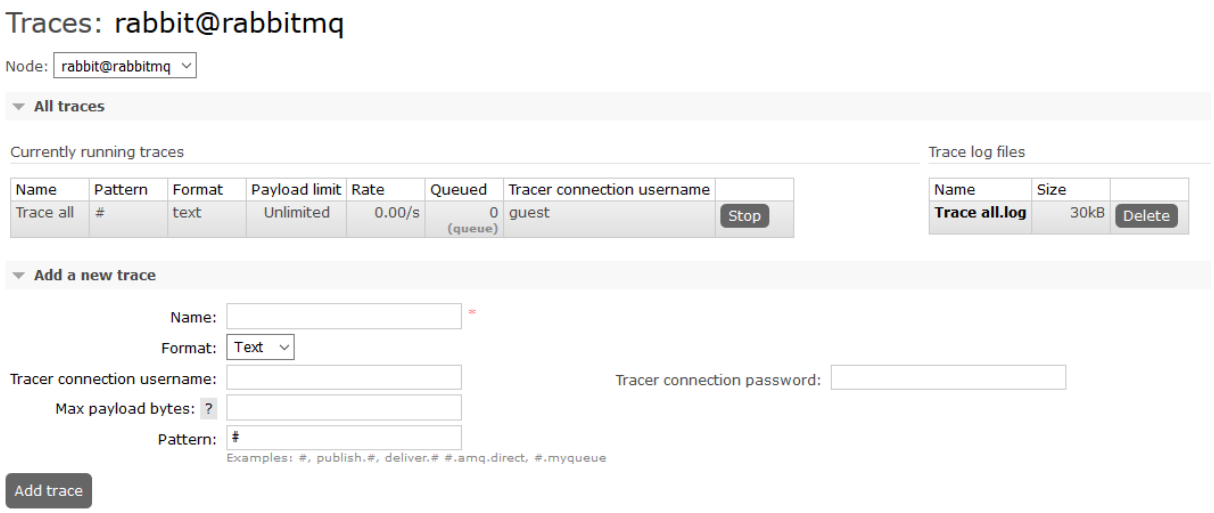

Abbildung 5 Firehose Konfiguration

5. Die Aufzeichnungen speichern die Nachrichten in Logdateien. Diese Dateien werden unter *All traces* angezeigt. Die Logdatei kann somit im Browser angezeigt werden. Beim Öffnen des Links muss man sich erneut mit guest / guest authentifizieren. Die Nachrichten werden dort aufgelistet. Abbildung 6 zeigt eine beispielhafte Nachricht.

```
================================================================================
2018-12-11 8:28:00:078: Message published

Node:         rabbit@rabbitmq
Connection:   172.24.0.10:59100 -> 172.24.0.2:5672
Virtual host: /
User:         guest
Channel:      1
Exchange:     ch.hslu.wipro.micros.Order
Routing keys: [<<"order.command.create">>]
Routed queues: [<<"ch.hslu.wipro.micros.OrderCreateCommand">>]
Properties:   [{<<"reply_to">>,longstr,<<"amq.gen-e3IRwyDCF4YzNhtzD6iXcA">>},
               {<<"correlation_id">>,longstr,
                <<"77ab4f1d-d8a2-4f63-95f4-7137b81bcb97">>}]
Payload:
{"customerId":1,"amountToArticle":{"23":1},"totalPrice":30.9,"status":"open"}
```

Abbildung 6 Firehose Nachricht

3.6 Lognachrichten von Containern abfragen

Diese Anleitung beschreibt wie auf die Lognachrichten von Containern zugegriffen werden kann.

Voraussetzungen

Das Application Lab wird lokal oder auf der virtuellen Maschine ausgeführt.

Vorgehen

1. Das Application Lab wird mit einem docker-compose Befehl ausgeführt. Daher ist es möglich, die Lognachriten von allen Containern auszulesen. Dazu muss ins Verzeichnis mit der docker-compose Konfigurationsdatei gewechselt werden. Anschliessend kann der Befehl `sudo docker-compose logs` abgesetzt werden.

2. Alternativ können die Logs auch nur von einem Container angezeigt werden. Dazu muss zunächst der CLI Command `sudo docker ps` ausgeführt werden. Dieser Befehl liefert eine Übersicht aller ausgeführter Container. Aus dieser Übersicht soll die Id des gewünschten Containers kopiert werden. Anschliessend können mit dem Befehl `sudo docker logs <CONTAINER_ID>` die Logs geholt werden.

3.7 Neuer Service einführen

Diese Anleitung beschreibt wie ein neuer Service in das Lab eingeführt werden kann.

Voraussetzungen

Das Application Lab Repository ist auf dem Client des Entwicklers geklont. Für den neuen Service muss bekannt sein, mit welchen anderen Services interagiert wird. Ausserdem muss bekannt sein, ob der Service einen eigenen Exchange definiert.

Vorgehen

1. Im neuen Service muss die Anbindung an den Message Broker definiert werden. Das heisst der Service muss seine Queues und allenfalls einen Exchange deklarieren. Dazu stellt RabbitMQ für viele Programmiersprachen Agents zur Verfügung.

   RabbitMQ-Host im Docker-Netzwerk: rabbitmq

2. Falls der neue Service Commands anbietet, können diese über den API Gateway angesprochen werden. Dazu im Java Gateway-Projekt ein neues Package mit dem Namen `ch.hslu.wipro.micros.service.SERVICENAME` erstellen.

3. Anschliessend müssen DTO-Klassen erstellt werden. Der Webservice soll später automatisch Objekte dieser Klassen erzeugen. Dies geschieht anhand der JSON-Nutzdaten der Requests, die an den Webservice verschickt werden.

4. Danach sollen vier Java-Klassen erstellt werden. Dafür werden folgende Namen empfohlen.

   ```
   <DOMAIN>MessageManager
   <DOMAIN>MessageDomainFactory
   <DOMAIN>CommandFactory
   <SERVICE>Service
   ```

   <DOMAIN> ist jeweils mit dem Domänennamen zu ersetzen.
   <SERVICE> ist jeweils mit dem Servicenamen zu ersetzen.

   Die Implementation dieser Klassen erfolgt aufgrund der bereits bestehenden Services.

5. Anschliessend sollte der Microservice über den API Gateway ansprechbar sein. In der Service-Klasse wird die Ressourcen URI definiert.
   http://localhost:7001/gateway/<RESSOURCE>

6. Falls der neue Service über den Webclient ansprechbar sein soll, muss auch die Website angepasst werden. Dazu ist eine neue Javascript Datei für den neuen Service zu erstellen. Diese Datei sollte im Verzeichnis `/application_lab/client/services` angelegt werden.

7. Zum Schluss muss die `index.html` Datei angepasst werden, so dass der neue Service auf der Website dargestellt wird.

3.8 Salesmanagement um einen Command erweitern

Diese Anleitung beschreibt wie der Salesmanagement, um einen neuen Command erweitert werden kann. Dies beinhaltet das Definieren eines neuen Topics, welches den neuen Routing Key und den neuen Queuenamen aneinanderbinden und das Erstellen eines neuen Consumers, um eingehende Commands auszuführen.

Voraussetzungen

Das Application Lab Repository ist auf dem Client des Entwicklers geklont.

Vorgehen

1. Im Salesmanagement muss ein neues Topic in der Klasse `CommandTopicsConsumerMap` im Modul `sm-business` definiert werden.

```
public CommandTopicsConsumerMap() {
    Topic <COMMAND_VAR> = new TopicBuilder()
        .atRoute("order.command.<COMMAND_NAME>")
        .atQueue("ch.hslu.wipro.micros.Order<COMMAND_NAME>Command")
        .build();

    handledTopics.put(<COMMAND_VAR>, <COMMAND_CONSUMER>.class);
    /…/
}
```

`<COMMAND_VAR>`	Eine Variable für den neuen Topic definieren.
`<COMMAND_NAME>`	Der Name des neuen Commands.
`<COMMAND_CONSUMER>`	Die Command Consumer Klasse, die das Command konsumieren soll.

2. Die `<COMMAND_CONSUMER>` Klasse im Package `rabbitmq/consumer` im Modul `sm-business` erstellen. Die Klasse muss eine Subklasse von DefaultConsumer sein und den folgenden Konstruktor besitzen. In der Methode `handleDelivery` kann die Nachricht konsumiert werden.

```
public class <COMMAND_CONSUMER> extends DefaultConsumer {
    public <COMMAND_CONSUMER> (Channel channel) {
        super(channel);
    }

    @Override
    public void handleDelivery(String consumerTag, Envelope envelope,
                               BasicProperties properties, byte[] body)
   { /…/ }
}
```

3. Das wars auch schon. Beim Start der Applikation wird die Erzeugung der Queues und das Binding mit den Routing Keys von der Klasse `CommandManager` übernommen. Diese ist im Modul `sm-business` im Packet `rabbitmq/manager` zu finden.

4 Tipps und bekannte Probleme

Die CLI-Befehle von Docker bieten einige Möglichkeiten, die in dieser Anleitung nicht abgedeckt werden. Die Dokumentation vom Hersteller ist aber gut gepflegt und kann unter dem folgenden Link abgerufen werden.
https://docs.docker.com/engine/reference/commandline/cli/

4.1 Problem: Docker Container lassen sich nicht mehr ausführen

Bei der lokalen Ausführung des Application Labs auf einem Windows Client konnten die Container teilweise nach einem Neustart nicht mehr ausgeführt werden. In einem solchen Fall kann versucht werden den Docker-Windows-Service manuell neu zu starten. Anschliessend sollten die bestehenden Container gelöscht werden.

`docker container prune`	Dieser Befehl löscht **alle** Container auf einem Host

Falls eine Ausführung immer noch nicht klappt, können zusätzlich auch alle Netzwerkinformationen und Images von Docker gelöscht werden. Die Images müssen bei der nächsten Ausführung wieder heruntergeladen werden. Der Build dauert dadurch deutlich länger.

`docker system prune –all`	Löscht sämtliche Images

5 Abbildungsverzeichnis