**Lab 6: Create a Service Spanning Multiple Containers**

Mou Zhang

## Section 1 Docker Network

1. Setting up the network pgnet as the doc suggest

docker network create pgnet

图形用户界面, 文本, 应用程序

描述已自动生成

## Section 2 Postgres Container

1. Pull the latest postgres docker

docker pull postgres

文本

描述已自动生成

1. Start a init postgres container with no table

docker run \

--network pgnet \

--name mypg \

-e POSTGRES\_PASSWORD=mysecret \

-d \

postgres

图形用户界面, 文本

描述已自动生成

## Section 3 Init Container

1. Get the ip address for the postgres container in pgnet

docker container inspect mypg \

-f '{{.NetworkSettings.Networks.pgnet.IPAddress}}'

文本

描述已自动生成

We can find that the ip address to connect to postgres is 172.18.0.2

1. Prepare the init.sql to initialize the pathcount database we need

CREATE TABLE IF NOT EXISTS pathcount (

path TEXT PRIMARY KEY,

count INT DEFAULT 0

);

电脑萤幕的截图

描述已自动生成

1. Setting up pathcount table in the postgres docker using another temporary docker with init.sql

docker run \

-i \

--rm \

--network pgnet \

-e PGPASSWORD=mysecret \

postgres \

psql -h 172.18.0.2 -U postgres < init.sql

文本

描述已自动生成

As you can see from the picture, the table is successfully created in the postgres conainter.

## Section 4 Service Container

1. Setting up the dockerfile and environment variable for service container

The ***Dockerfile*** is shown as below:

FROM python:3.6

RUN mkdir /app

ADD . /app

WORKDIR /app

RUN pip install flask \

Psycopg2

EXPOSE 8080

ENV DB\_NAME="postgres"

ENV DB\_USER="postgres"

ENV DB\_HOST="172.18.0.2"

ENV DB\_PASSWORD="mysecret"

ENV FLASK\_APP=main.py

CMD flask run --host=0.0.0.0 --port=8080

As you can see, It is using python 3.6 docker and using flask and psycopg2 to set up the service and gain access to the postgres docker. The postgres username, database, password, and hostname are configured in the environment variables.

1. Setting up the frontend and backend of the service

The ***main.py*** is the backend service. The backend service is using flask to get configurations from environment and connect to the database and add count/return count to the frontend with render\_templates using jinja2 format.

# Copyright 2018 Google LLC

#

# Licensed under the Apache License, Version 2.0 (the "License");

# you may not use this file except in compliance with the License.

# You may obtain a copy of the License at

#

# http://www.apache.org/licenses/LICENSE-2.0

#

# Unless required by applicable law or agreed to in writing, software

# distributed under the License is distributed on an "AS IS" BASIS,

# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

# See the License for the specific language governing permissions and

# limitations under the License.

# [START gae\_python38\_app]

from flask import Flask, render\_template

import psycopg2

import os

app = Flask(\_\_name\_\_)

dbname = os.getenv('DB\_NAME')

dbuser = os.getenv('DB\_USER')

# "localhost" # "pgnet" # "172.18.0.2" # os.getenv('DB\_HOST')

dbhost = os.getenv('DB\_HOST')

dbpasswd = os.getenv('DB\_PASSWORD')

@app.route('/', defaults={'u\_path': ''}, methods=['GET'])

@app.route('/<path:u\_path>', methods=['GET'])

def root(u\_path):

count\_paths(str(u\_path))

return show\_path()

def count\_paths(u\_path):

sql = """INSERT INTO pathcount (path, count)

VALUES (%s, 1)

ON CONFLICT (path) DO UPDATE

SET count = pathcount.count + 1

RETURNING count;"""

print(u\_path)

try:

conn = psycopg2.connect(

database=dbname, user=dbuser, host=dbhost, password=dbpasswd)

cur = conn.cursor()

cur.execute(sql, (u\_path, ))

conn.commit()

cur.close()

conn.close()

except psycopg2.DatabaseError as e:

print(e)

print("I am unable to connect to the database.")

def show\_path():

sql = """SELECT path, count FROM pathcount ORDER BY path;"""

data\_return = None

try:

conn = psycopg2.connect(

database=dbname, user=dbuser, host=dbhost, password=dbpasswd)

cur = conn.cursor()

cur.execute(sql)

data\_return = cur.fetchall()

conn.commit()

cur.close()

conn.close()

except psycopg2.DatabaseError as e:

print(e)

print("I am unable to connect to the database.")

print(data\_return)

return render\_template('index.html', data=data\_return)

if \_\_name\_\_ == '\_\_main\_\_':

# This is used when running locally only. When deploying to Google App

# Engine, a webserver process such as Gunicorn will serve the app. This

# can be configured by adding an `entrypoint` to app.yaml.

app.run(host='127.0.0.1', port=8080, debug=True)

# [END gae\_python38\_app]

1. Setting up the frontend part with ***index.html***. Using jinja2 to get and split data, then show the data on the web page in a html table.

<!DOCTYPE>

<html>

<title>

Mou's Count Path

</title>

<table border = 1>

<tr><th>Path</th><th>Count</th></tr>

{%for i in data%}

<tr> <td>{{i[0]}}</td> <td>{{i[1]}}</td> </tr>

{%endfor%}

</table>

</html>

1. Build the service docker. Naming it as lab6-img

docker build -t lab6-img .

11.文本

描述已自动生成

11. Build the service container with lab6-img and connect it to localhost:8080

docker run \

-it \

--name=path\_count \

--network pgnet\

-p 8080:8080 \

lab6-img

文本

描述已自动生成

1. Then you can access the service with localhost:8080. The table is shown as expected, with all path and count sorted and printed.

图形用户界面, 应用程序

描述已自动生成

If you access a path already in the database, like ‘localhost:8080/test’, then the count for path ‘test’ add 1.

图形用户界面, 应用程序

描述已自动生成

If you access a path that is not in the database, like ‘localhost:8080/newtest’, then that path is add to database and the count is set to one.

图形用户界面, 应用程序, Word

描述已自动生成

As you can see in the pictures, all functions are implemented correctly. The lab is completed.

## Section 5 Answer to some security questions

* **Why did we create a special network instead of exposing the host network?**

Because this prevents us from exposing the database to the public network, which is dangerous and easy to hack. This network provides an isolation between the container and the host network.

* **Why didn’t we use exposed ports everywhere (that they exist)?**

Because it’s not secure. If we exposed ports everywhere, then these ports are easily reached from the host network, which gives hackers chances to hack into them and get data. By using another network, we get an isolation and protection from these hackers and malwares.

* **What could happen if you didn’t use SQL parameters, but relied on string formatting for setting the path in your queries?**

Then it is easy to get SQL injection attack.

* **Why is that particularly important in this setup? What makes those parameters potentially dangerous?**

Because this is helping prevent from the risky parameters. These parameters still have potential problems if not carefully checked.

* **The bridge network we define only works on a single host. What would you have to do to make these containers talk to each other if they were running on *different host machines*?**

One possible way is to set up a bridge network similar to this network and try to communicate through this network. Another possible way is to use more advanced structure like docker compose and Kubernetes.

* **What parts of this did you wish were simpler? Which parts seemed unnecessarily difficult?**

I think the part that should be simpler is the part with writing service docker and connect with postgres docker. Since this is a lab for security not backend developing, I think we shall get more hints on the service container building, including the suggested packages.

The part seems to be unnecessarily difficult is the conflict with postgres docker and pgadmin on my local macbook. When I was trying to connect to the postgres docker using port 5432, this port is occupied on my local machine and there is no reminder on what is the reason. don’t find the specific reason for the conflict, but it is really frustrating since I tried so many times of connection and don’t find the reason for failure. This problem only be solved after I completely removed all my postgres on the macbook and it takes a lot of time to realize the real problem.