Lab 6

Ziyang Lin zlin32@jhu.edu

Setting-up

1. Create Docker network
   Command: docker network create pgnet

```
PS E:\isi\cloud computing\cloud_computing_lab\lab6> docker network create pgnet
dca58e29d8863427a9a913c94f3b4d52e916165da70a025a8d19fe3f834b2f82
```

2. Create Postgres Container
   2.1 Pull the Postgres image from repositories.
   Command: docker pull postgres

```
PS E:\isi\cloud computing\cloud_computing_lab\lab6> docker pull postgres
Using default tag: latest
latest: Pulling from library/postgres
bb79b6b2107f: Pull complete
e3dc51fa2b56: Pull complete
f213b6f96d81: Pull complete
2780ac832fde: Pull complete
ae5cee1a3f12: Pull complete
95db3c06319e: Pull complete
475ca72764d5: Pull complete
8d602872ecae: Pull complete
c4fca31f2e3d: Pull complete
a00c442835e0: Pull complete
2e2305af3390: Pull complete
6cff852bb872: Pull complete
25bb0be11543: Pull complete
4738c099c4ad: Pull complete
Digest: sha256:8f7c3c9b61d82a4a021da5d9618faf056633e089302a726d619fa467c73609e4
Status: Downloaded newer image for postgres:latest
docker.io/library/postgres:latest
```

   2.2 Run container using Postgres image. Connect it to the network pgnet
   Command: docker run –network pgnet –name pg1 \
   -e POSTGRES_PASSWORD=secret -d postgres

```
PS E:\isi\cloud computing\cloud_computing_lab\lab6> docker run --network pgnet --name pg1
-e POSTGRES_PASSWORD=secret -d postgres
fd0ba98dfaf649d04a525f6cf7c9889981837c73152f089f59fc030182a9e36a
```

3. Init Postgres Container
   3.1 Find the IP address of postgres database in container.
   Command: docker container inspect pg1 \
   -f '{{.NetworkSettings.Networks.pgnet.IPAddress}}'

```
PS E:\isi\cloud computing\cloud_computing_lab\lab6> docker container inspect pg1 -f
'{{.NetworkSettings.Networks.pgnet.IPAddress}}'
172.18.0.3
```

   3.2 Use init.sql to build table in database.

```
PS E:\isi\cloud computing\cloud_computing_lab\lab6> cat init.sql
CREATE TABLE IF NOT EXISTS pathcount (
  path TEXT PRIMARY KEY,
  count INT DEFAULT 0
);
```

3.3 Use init.sql to initialize database.

Command: Get-Content init.sql | docker run -it –rm –network pgnet \

-e PGPASSWORD=secret postgres psql -h 172.18.0.3 -U postgres

But I got an 'Input device is not a TTY' error.

```
PS E:\isi\cloud computing\cloud_computing_lab\lab6> Get-Content init.sql | docker ru
n -it --rm --network pgnet -e PGPASSWORD=secret postgres psql -h 172.18.0.3 -U postg
res
the input device is not a TTY.  If you are using mintty, try prefixing the command w
ith 'winpty'
```

Command: Get-Content init.sql | docker run -i –rm –network pgnet \

-e PGPASSWORD=secret postgres psql -h 172.18.0.3 -U postgres

Using -i can work.

```
PS E:\isi\cloud computing\cloud_computing_lab\lab6> Get-Content init.sql | docker ru
n -i --rm --network pgnet -e PGPASSWORD=secret postgres psql -h 172.18.0.3 -U postgr
es
NOTICE:  relation "pathcount" already exists, skipping
CREATE TABLE
```

Create sevice Container

1. Package requirements
   I wrote my web application by using Python and Flask. And I need to install Flask for creating application, psycopg2 for working with postgresql database, and pyhocon to handle environment variables.
2. Environment Variables
   I defined all configuration variables in environment variables. And setting the variables in db.conf file, which will be loaded in application.

```
PS E:\isi\cloud computing\cloud_computing_lab\lab6> cat .\db.conf
 databases {
    "postgres" = {
      host = 172.18.0.3
      user = "postgres"
      database = "postgres"
      password = "secret"
    }
  }
```

3. Code Files
   I wrote application file in main.py and HTML file in index.html

4. Then I built my container using Dockerfile.
   Command: docker build -t pathcount .

```
PS E:\isi\cloud computing\cloud_computing_lab\lab6> cat .\Dockerfile
FROM python
RUN mkdir /app
COPY . /app/
WORKDIR /app
RUN pip install Flask \
  pyhocon \
  psycopg2
EXPOSE 8080
CMD ["python", "main.py"]
```

```
PS E:\isi\cloud computing\cloud_computing_lab\lab6> docker build -t pathcount .
[+] Building 181.1s (10/10) FINISHED
```

5. Then ran service container in port 10080.
   Command: docker run -it --name=pc_container0 –network pgnet -p 10080:8080 pathcount

```
PS E:\isi\cloud computing\cloud_computing_lab\lab6> docker run -it --name=pc_container0 --network pgnet -p 10080:8080 pa
thcount
 * Serving Flask app "main" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 181-155-360
```

6. Finally, go to localhost:10080 on browser and the page show like this.

Questions

- **Why did we create a special network instead of exposing the host network?**
  It can isolate the database from host network, which can ensure data security.
- **Why didn't we use exposed ports everywhere (that they exist)?**
  It use docker bridge network to isolate the database in postgres container. If we use exposed ports, the database may be exposed in outside attacks.
- **What could happen if you didn't use SQL parameters, but relied on string formatting for setting the path in your queries?**

  The application will be easily attacked by SQL injection attacks.

- **Why is that particularly important in this setup? What makes those parameters potentially dangerous?**
  The parameters didn't be filtered and checked when running the container. So there are potential threats that parameter can be used in injection attacks.
- **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?**
  If the containers are running on different host machines, I would use the host network like I did in this lab.

- **What parts of this did you wish were simpler? Which parts seemed unnecessarily difficult?**
  The most difficult things I met in this lab is that I firstly tried to set up the container in VirtualBox, however I didn't left enough disk space for this lab. So I have to delete file but still no enough space. I hope if possible we can put the space requirements in the lab document, then I will not waste so much time. And the -it stuff in instrument couldn't use in lab. I think that is unnecessarily difficult.