BTS | Barcelona Technology School

Docker compose exercise

Docker may be introduced first using only the Docker engine CLI which is a powerful tool for creating infrastructures from already existent Docker images. Docker engine CLI also provides some advanced features as volumes or ports mapping.
You can clone the git repository:
git clone https://github.com/BTSruben/3_BDI_18mar_Network_Assignment

Creating a Dockerfile

Creating a new Docker image defined through a Dockerfile is not a difficult task checking out the official Docker documentation for Dockerfiles.
Next a basic Dockerfile is defined, compiled and executed. This sample Dockerfile creates a python container containing a PostgreSQL client (psycopg2) and Flask dependencies.

Dockerfile:

```
FROM python:3.7

# Initialize
WORKDIR
COPY requirements.txt /root/

# Setup
RUN pip3 install --upgrade pip
RUN pip3 install -r requirements.txt

EXPOSE
```

Requirements.txt

```
Flask
psycopg2
Flask-SQLAlchemy
```

Dockerfile inherits from a Python Docker image (version 3.7).
The FROM instruction adds into this Dockerfile everything within de Python 3.7 Docker image.
We define our WORKDIR (By default is the root path /, but we'll change to root user folder)
Requirements.txt file contains our python dependecies and we'll copy it inside the image to install them after.
RUN commands will install our dependencies using de command pip
We'll use this image to create a web application with Flask and FlaskApp (our webserver) will use the port 5000 to listen the requests.

Docker compose

Docker Compose is a tool for defining and running multi-container Docker applications. You use a YAML file (docker-compose.yml) to configure your application's services. Then you catn create and start all the services from your configuration with a single command.

In the same folder of docker-compose.yml:

Start infrastructure:

```
$ docker-compose up
```

Start Background mode:

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

Stop infrastructure:

```
$ docker-compose down
```

Next sample defines a Docker compose file able to run two containers. One PostgreSQL container and a python container (the one we have previously defined)

```
version: '3'
services:
  app:
    build: .
    hostname: app_flask
    container_name: app_flask
    restart: always
    volumes:
     - .:/root/my_code
    stdin_open: true
    ports:
     - "5000:5000"
    tty: true
    networks:
        - db_backend
  db:
    image: postgres:11
    hostname: db_postgres
    container_name: db_postgres
    restart: always
    environment:
      POSTGRES_PASSWORD: Mypass.1234
    networks:
     - db_backend

networks:
  db_backend:
    driver: bridge
```

Syntax of docker-compose.yml

version (version of the docker compose, version 3 is the latest version 3-2019)

services (After the declaration we must define the services, our services are **app** and **db**)

build (If we want to build our own image, we can do it with this option. Remember the character . means current path, the same path of the file)

*In the case of the service **db** we'll use a public image called **postgres** with the version **11** using the option **image.***

*So we have 2 ways, **build: <image_path>** or **image: <image_name>***

hostname/container_name (name of the container)

volumes (defines shared folders https://docs.docker.com/compose/compose-file/#volume-configuration-reference)

stdin_open/tty (keeps stdin open to enter command and terminal is opened connecting stdin)

ports (We can connect a port of the host and a port of the container)

If you want to send a request to a server inside a container, you should create a link between the host port and the container port. For example, "1234:5000" would be URL: <localhost:1234> (HOST machine) connected to <localhost:5000> (CONTAINER machine)

So, If you start a server inside the **container** and this server is listening the port **5000**, you can open your browser in the **host machine** and use the URL: localhost:**1234**. This URL will be linked to container port **5000**.

restart (always option restarts the container if any error stops it)

environment (define environment variables)
Extra info: https://www.digitalocean.com/community/tutorials/how-to-read-and-set-environmental-and-shell-variables-on-a-linux-vps

networks (After the declaration we must define the networks, we'll create a network called **db_backend**, the type will be bridge https://docs.docker.com/network/bridge/)

Extra information of Docker compose: https://docs.docker.com/compose/

If we list the files of our project folder: (Host machine)

```
$ ls -l
total 1784
-rw-r--r--  1 nimbo4  staff     163 Mar 19 18:35 Dockerfile
-rw-r--r--  1 nimbo4  staff     120 Mar 19 19:10 README.md
-rw-r--r--  1 nimbo4  staff    1394 Mar 19 18:54 app.py
-rw-r--r--  1 nimbo4  staff     490 Mar 19 23:40 docker-compose.yml
-rw-r--r--  1 nimbo4  staff      32 Mar 19 18:35 requirements.txt
drwxr-xr-x  4 nimbo4  staff     128 Mar 19 18:03 static
drwxr-xr-x  4 nimbo4  staff     128 Mar 19 19:07 templates
```

Description of the files:

Dockerfile	Docker configuration file
README.md	Description of the project
app.py	Code in python of our web application in Flask
docker-compose.yml	Docker compose config. file
requirements.txt	Python dependencies
static	Static files. Web application needs them
templates	HTML files. Web application needs them

Now, we will start the infrastructure. Your output should be similar to mine.

Execution of the project:

```
nimbo4:3_BDI_18mar_Network_Assignment nimbo4$ docker-compose up
Creating network "3_bdi_18mar_network_assignment_db_backend" with driver "bridge"
Building app
Step 1/6 : FROM python:3.7
 ---> 32260605cf7a
Step 2/6 : WORKDIR /root
 ---> Using cache
 ---> de00314d2049
Step 3/6 : COPY requirements.txt /root/
 ---> b4d93f1f3d02
Step 4/6 : RUN pip3 install --upgrade pip
 ---> Running in f60ef2a39922
Requirement already up-to-date: pip in /usr/local/lib/python3.7/site-packages (19.0.3)
Removing intermediate container f60ef2a39922
 ---> 60ba29547ab6
Step 5/6 : RUN pip3 install -r requirements.txt
 ---> Running in 024e6608ad38
Collecting Flask (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/7f/
e7/08578774ed4536d3242b14dacb4696386634607af824ea997202cd0edb4b/Flask-1.0.2-py2.py3-none-any.whl
(91kB)
Collecting psycopg2 (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/0c/ba/
e521b9dfae78dc88d3e88be99c8d6f8737a69b65114c5e4979ca1209c99f/psycopg2-2.7.7-cp37-cp37m-
manylinux1_x86_64.whl (2.7MB)
Collecting Flask-SQLAlchemy (from -r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/
a1/44/294fb7f6bf49cc7224417cd0637018db9fee0729b4fe166e43e2bbb1f1c8/Flask_SQLAlchemy-2.3.2-py2.py3-
none-any.whl
Collecting itsdangerous>=0.24 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/76/ae/
44b03b253d6fade317f32c24d100b3b35c2239807046a4c953c7b89fa49e/itsdangerous-1.1.0-py2.py3-none-
any.whl
Collecting click>=5.1 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/fa/
37/45185cb5abbc30d7257104c434fe0b07e5a195a6847506c074527aa599ec/Click-7.0-py2.py3-none-any.whl
(81kB)
Collecting Jinja2>=2.10 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/7f/ff/
ae64bacdfc95f27a016a7bed8e8686763ba4d277a78ca76f32659220a731/Jinja2-2.10-py2.py3-none-any.whl
(126kB)
Collecting Werkzeug>=0.14 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/29/5e/
d54398f8ee78166d2cf07e46d19096e55aba506e44de998a1ad85b83ec8d/Werkzeug-0.15.0-py2.py3-none-any.whl
(328kB)
Collecting SQLAlchemy>=0.8.0 (from Flask-SQLAlchemy->-r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/85/29/
d7a5687d0d21ea8133f2d4ef02dfb4d191afe7ebc8bd9f962d99bdf595e1/SQLAlchemy-1.3.1.tar.gz (5.9MB)
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10->Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/98/7b/
ff284bd8c80654e471b769062a9b43cc5d03e7a615048d96f4619df8d420/MarkupSafe-1.1.1-cp37-cp37m-
manylinux1_x86_64.whl
Building wheels for collected packages: SQLAlchemy
  Building wheel for SQLAlchemy (setup.py): started
  Building wheel for SQLAlchemy (setup.py): finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/d6/22/ef/
dc48efbf5df0c3b73cc6d6ccc682c27a8ecbf072488aea9eaa
Successfully built SQLAlchemy
Installing collected packages: itsdangerous, click, MarkupSafe, Jinja2, Werkzeug, Flask, psycopg2,
SQLAlchemy, Flask-SQLAlchemy
Successfully installed Flask-1.0.2 Flask-SQLAlchemy-2.3.2 Jinja2-2.10 MarkupSafe-1.1.1
SQLAlchemy-1.3.1 Werkzeug-0.15.0 click-7.0 itsdangerous-1.1.0 psycopg2-2.7.7
Removing intermediate container 024e6608ad38
 ---> 7364f919b639
Step 6/6 : EXPOSE 5000
 ---> Running in 2af6484499cf
Removing intermediate container 2af6484499cf
 ---> b12a61568e08
Successfully built b12a61568e08
Successfully tagged 3_bdi_18mar_network_assignment_app:latest
```

```
Creating app_flask   ... done
Creating db_postgres ... done
Attaching to db_postgres, app_flask
db_postgres | The files belonging to this database system will be owned by user
"postgres".
db_postgres | This user must also own the server process.
db_postgres |
db_postgres | The database cluster will be initialized with locale "en_US.utf8".
db_postgres | The default database encoding has accordingly been set to "UTF8".
db_postgres | The default text search configuration will be set to "english".
db_postgres |
db_postgres | Data page checksums are disabled.
db_postgres |
app_flask | Python 3.7.2 (default, Mar  5 2019, 06:22:51)
app_flask | [GCC 6.3.0 20170516] on linux
app_flask | Type "help", "copyright", "credits" or "license" for more information.
db_postgres | fixing permissions on existing directory /var/lib/postgresql/data ... ok
db_postgres | creating subdirectories ... ok
db_postgres | selecting default max_connections ... 100
db_postgres | selecting default shared_buffers ... 128MB
db_postgres | selecting dynamic shared memory implementation ... posix
db_postgres | creating configuration files ... ok
db_postgres | running bootstrap script ... ok
db_postgres | performing post-bootstrap initialization ... ok
db_postgres | syncing data to disk ... ok
db_postgres |
db_postgres | WARNING: enabling "trust" authentication for local connections
db_postgres | You can change this by editing pg_hba.conf or using the option -A, or
db_postgres | --auth-local and --auth-host, the next time you run initdb.
db_postgres |
db_postgres | Success. You can now start the database server using:
db_postgres |
db_postgres |     pg_ctl -D /var/lib/postgresql/data -l logfile start
db_postgres |
db_postgres | waiting for server to start....2019-03-19 18:36:57.374 UTC [41] LOG:
listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db_postgres | 2019-03-19 18:36:57.394 UTC [42] LOG:  database system was shut down at
2019-03-19 18:36:54 UTC
db_postgres | 2019-03-19 18:36:57.401 UTC [41] LOG:  database system is ready to
accept connections
db_postgres |
db_postgres | server started
db_postgres |
db_postgres | /usr/local/bin/docker-entrypoint.sh: ignoring /docker-entrypoint-
initdb.d/*
db_postgres |
db_postgres | waiting for server to shut down...2019-03-19 18:36:57.447 UTC [41] LOG:
received fast shutdown request
db_postgres | .2019-03-19 18:36:57.450 UTC [41] LOG:  aborting any active transactions
db_postgres | 2019-03-19 18:36:57.453 UTC [41] LOG:  background worker "logical
replication launcher" (PID 48) exited with exit code 1
db_postgres | 2019-03-19 18:36:57.456 UTC [43] LOG:  shutting down
db_postgres | 2019-03-19 18:36:57.476 UTC [41] LOG:  database system is shut down
db_postgres |
db_postgres | server stopped
db_postgres |
db_postgres | PostgreSQL init process complete; ready for start up.
db_postgres |
db_postgres | 2019-03-19 18:36:57.570 UTC [1] LOG:  listening on IPv4 address
"0.0.0.0", port 5432
db_postgres | 2019-03-19 18:36:57.570 UTC [1] LOG:  listening on IPv6 address "::",
port 5432
db_postgres | 2019-03-19 18:36:57.575 UTC [1] LOG:  listening on Unix socket "/var/
run/postgresql/.s.PGSQL.5432"
db_postgres | 2019-03-19 18:36:57.593 UTC [50] LOG:  database system was shut down at
2019-03-19 18:36:57 UTC
db_postgres | 2019-03-19 18:36:57.600 UTC [1] LOG:  database system is ready to accept
connections
```

Extra information regarding the usage of Docker compose may be found in to the official docs.

If you have send the command ‘ docker-compose up ‘, the terminal is used to show us the log of the containers, so we can’t use this terminal, we need to open another.

If we open another terminal in our host machine, we can see our containers running using ‘docker ps’

```
$ docker ps
CONTAINER ID   IMAGE           COMMAND                          CREATED        STATUS
PORTS        NAMES
e8772b7a026d   postgres:11     "docker-entrypoint.s…"           4 hours ago    Up 4 hours
5432/tcp                db_postgres
90097d3f8c37   3_bdi_18mar_network_assignment_app   "python3"  4 hours ago    Up 4 hours
0.0.0.0:5000->5000/tcp   app_flask
```

Remind the option volume in the service (container) app:

```
volumes:
      - .:/root/my_code
```

We can check if our files are inside the container called **app_flask**, I know it’s called app_flask because we defined `container_name: app_flask` in the docker compose file.

Note I’m using the container name instead of the id container:

```
$docker exec -it app_flask bash
root@app_flask:~#
```

Now, I’ve logged in my **app_flask** container with the user **root**.

My shared folder was /root/my_code: (These commands are in container machine)

```
root@app_flask:~# ls -l /root/
total 4
drwxr-xr-x 12 root root 384 Mar 19 23:00 my_code
-rw-r--r--  1 root root  32 Mar 19 17:35 requirements.txt
root@app_flask:~# ls -l /root/my_code/
total 904
-rw-r--r-- 1 root root    163 Mar 19 17:35 Dockerfile
-rw-r--r-- 1 root root    120 Mar 19 18:10 README.md
-rw-r--r-- 1 root root   1394 Mar 19 17:54 app.py
-rw-r--r-- 1 root root    490 Mar 19 22:40 docker-compose.yml
-rw-r--r-- 1 root root     32 Mar 19 17:35 requirements.txt
drwxr-xr-x 4 root root    128 Mar 19 17:03 static
drwxr-xr-x 4 root root    128 Mar 19 18:07 templates
```

We can see the same files of our project shared in the path /root/my_code inside the container.

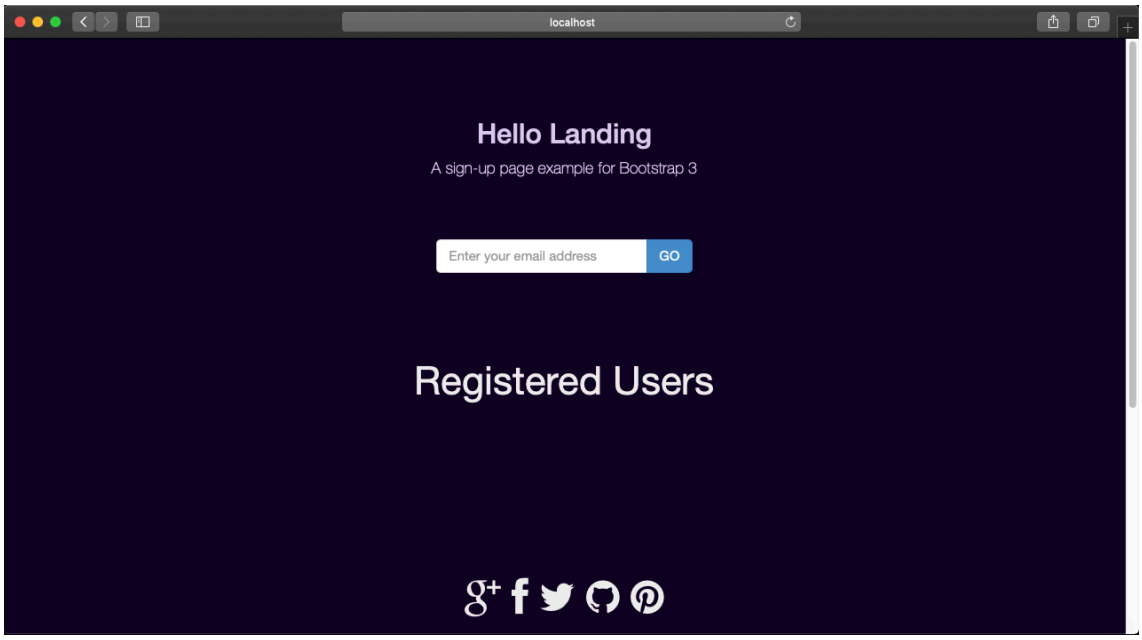
Now, we will execute our web application in python.

(Container machine)

```
root@app_flask:~# cd /root/my_code/
root@app_flask:~/my_code# python app.py
/usr/local/lib/python3.7/site-packages/flask_sqlalchemy/__init__.py:
794: FSADeprecationWarning: SQLALCHEMY_TRACK_MODIFICATIONS adds
significant overhead and will be disabled by default in the future.
Set it to True or False to suppress this warning.
  'SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and '
/usr/local/lib/python3.7/site-packages/psycopg2/__init__.py:144:
UserWarning: The psycopg2 wheel package will be renamed from release
2.8; in order to keep installing from binary please use "pip install
psycopg2-binary" instead. For details see: <http://initd.org/psycopg/
docs/install.html#binary-install-from-pypi>.
  """)
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production
environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
/usr/local/lib/python3.7/site-packages/flask_sqlalchemy/__init__.py:
794: FSADeprecationWarning: SQLALCHEMY_TRACK_MODIFICATIONS adds
significant overhead and will be disabled by default in the future.
Set it to True or False to suppress this warning.
  'SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and '
/usr/local/lib/python3.7/site-packages/psycopg2/__init__.py:144:
UserWarning: The psycopg2 wheel package will be renamed from release
2.8; in order to keep installing from binary please use "pip install
psycopg2-binary" instead. For details see: <http://initd.org/psycopg/
docs/install.html#binary-install-from-pypi>.
  """)
 * Debugger is active!
 * Debugger PIN: 576-107-130
```

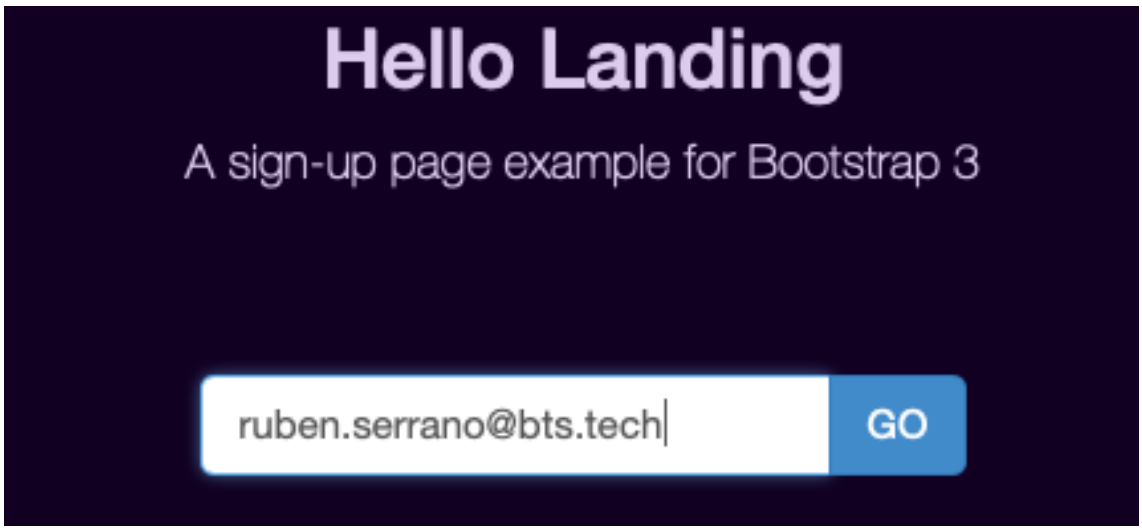
Again, the terminal is used to show us a log, in this case the Flask output.

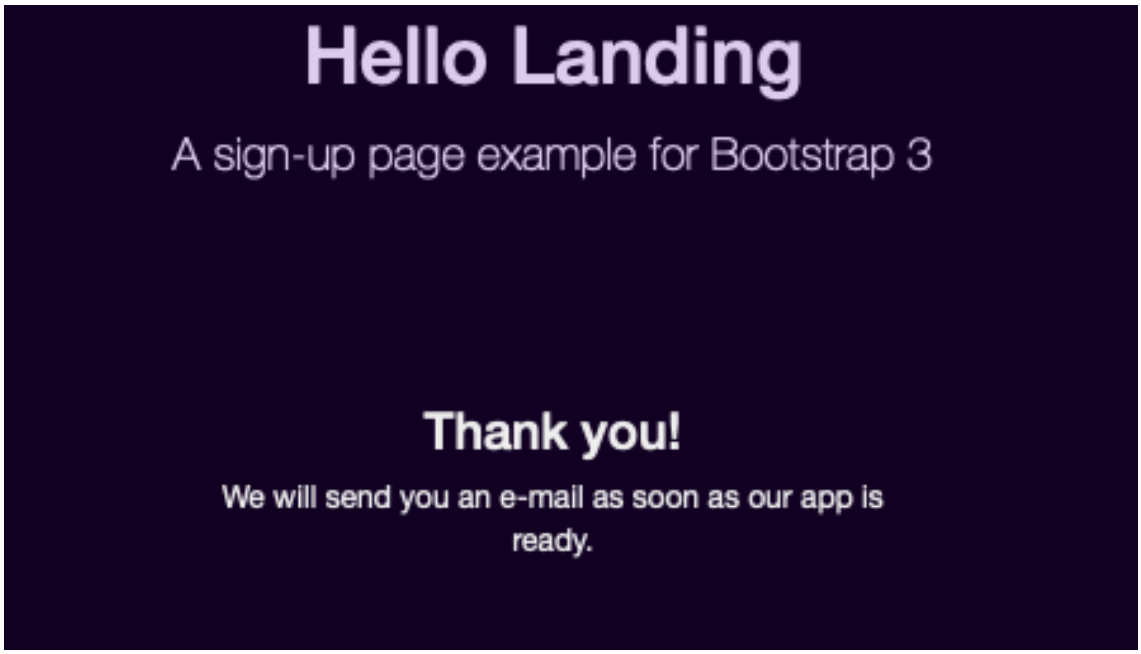
Now, If you have the containers running and the Flask script running (python app.py), you can use a browser (Host machine) and go to localhost:5000



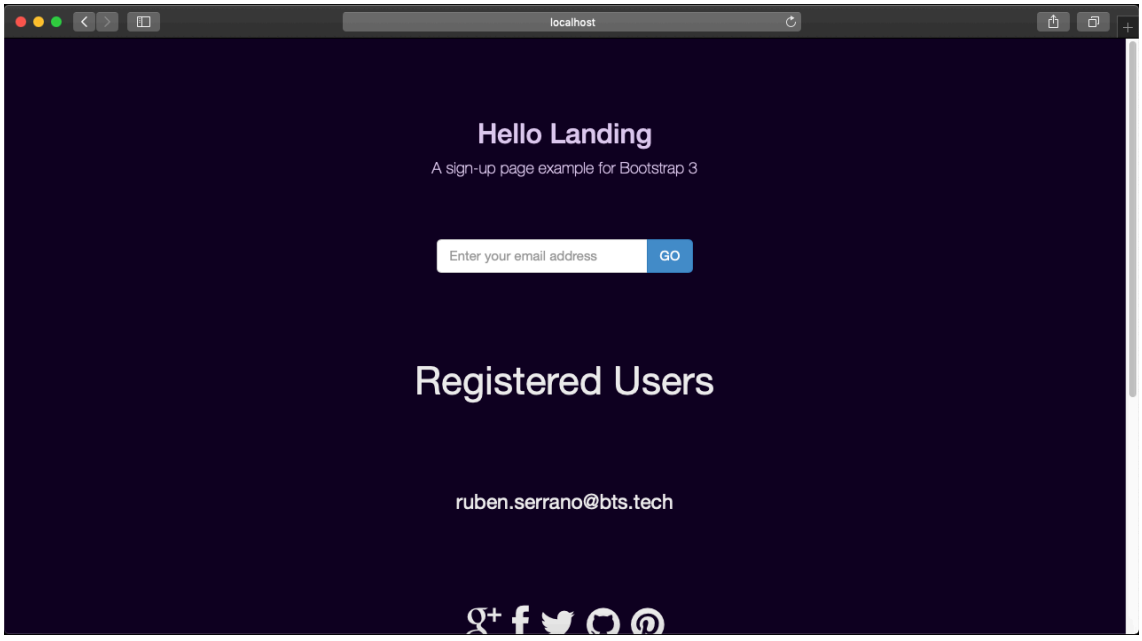
The application is a sample of how can we use Flask to insert mails into PostgresSQL and how to show them.

If you write an email and click go. A register will be done.





Finally, If you return to localhost:5000, you will see the mail list:



1. Reproduce all these steps in your host machine.
Copy the output in a file. (8 points)

2. Design your own docker-compose.yml.
Requirements: at least one service and one shared volume.
Copy the output in a file. (2 points)