ALPEN-ADRIA
UNIVERSITÄT
KLAGENFURT

Building a simple Python web application

Narges Mehran, MSc.

Current Topics in Distributed Systems: Internet of Things and Cloud Computing,

SS2021

26.05.2021

What do you need?

- ➢ Ubuntu OS
- ➢ Redis (Remote Dictionary Server)
  - ▪ is an in-memory data structure project implementing a distributed, in-memory key-value database with optional durability.
- ➢ Flask is a micro-web framework written in Python

Creating an Ubuntu based container

Download an *image*

```
    docker pull ubuntu:latest
```

Create and run *container* from *image*

```
    docker create -t --name erst -p 5000:5000
ubuntu:latest
```

Start the *container*

```
    docker start erst
```

Check if *container* is running

```
    docker ps
    docker ps -a
```

Setup a container from a Docker image

```
docker exec -it erst /bin/bash
#tty into the container
   • apt-get update
     #update repos
   • apt-get install -y python3
     #install python
   • apt install -y python3-pip
     #install python package manager
   • apt-get install -y redis-server
     #install redis-server
   • systemctl enable redis-server.service
     #enable Redis to start on system boot
```

Setup a container from a Docker image

Problem with systemctl?

1. Run:
   ```
   apt-get install systemd
   /lib/systemd/systemd-sysv-install enable redis-
   server
   ```
2. Open this file with your preferred text editor:
   ```
   nano /etc/redis/redis.conf
   ```
   Inside the file, find the `supervised` directive. This directive allows you to declare an init system to manage Redis as a service, providing you with more control over its operation. The `supervised` directive is set to `no` by default. Since you are running Ubuntu, which uses the systemd init system, change this to `systemd`.

Setup a container from a Docker image

Verifying

- *python3 –version*
- *pip3 –version*

Installing the following dependencies

- *pip3 install flask redis*

Setting some *env. variables* for flask

- *echo export LC_ALL=C.UTF-8 >> /root/.bashrc*
- *echo export LANG=C.UTF-8 >> /root/.bashrc*
- *echo export FLASK_APP=/root/app.py >> /root/.bashrc*
- *echo export FLASK_RUN_HOST=0.0.0.0 >> /root/.bashrc*
- *exit*

Setup an application

- ✓ Copy the app.py local -> container
  - *docker cp ~/app.py erst:/root/app.py*
- ✓ app.py is this code →
- ✓ tty into the container again
- ✓ Run the application
  - *flask run*

app.py

```
import time

import redis
from flask import Flask

app = Flask(__name__)
cache = redis.Redis(host='localhost', port=6370)


def get_hit_count():
   retries = 5
   while True:
      try:
         return cache.incr('hits')
      except redis.exceptions.ConnectionError as exc:
         if retries == 0:
            raise exc
         retries -= 1
         time.sleep(0.5)


@app.route('/')
def hello():
   count = get_hit_count()
   return 'Hello World! I have been seen {}
times.\n'.format(count)
```

Run the application

Open a terminal and execute:

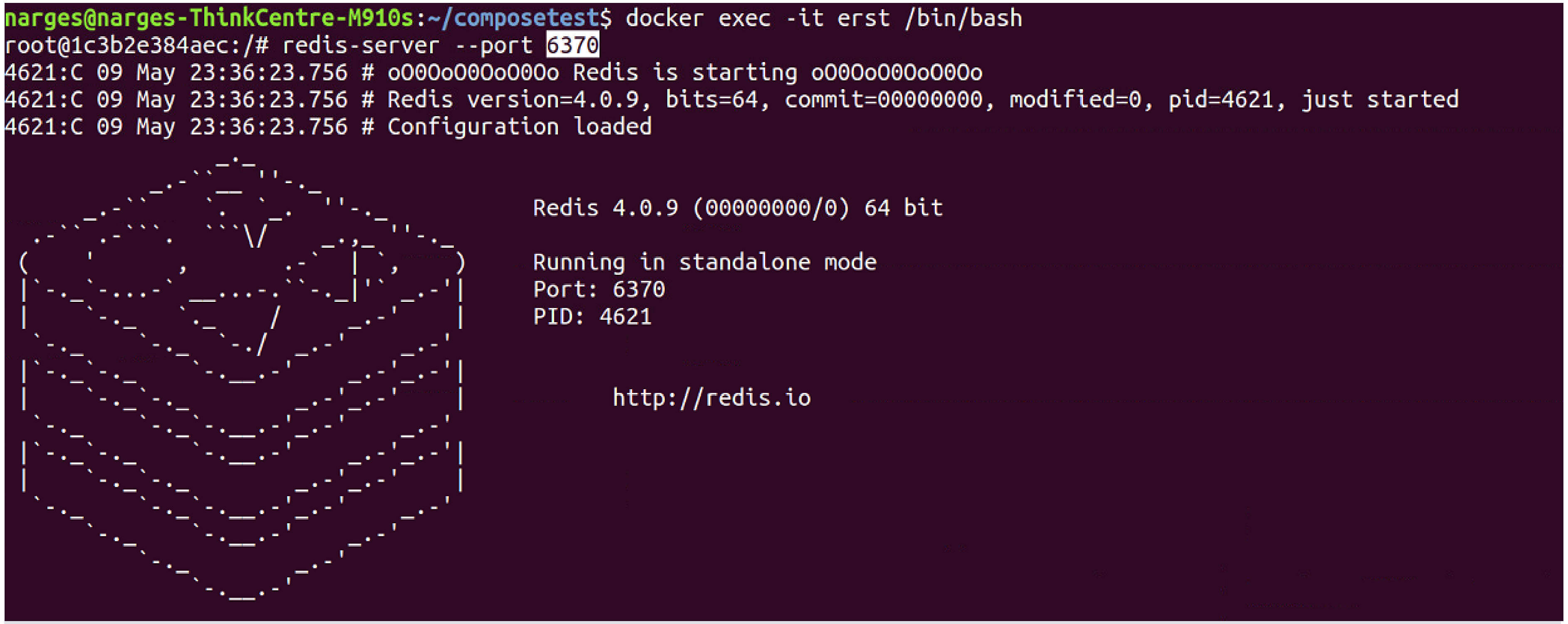
```
docker exec -it erst /bin/bash
• redis-server --port 6370
```

Open another terminal and execute:

```
docker exec -it erst /bin/bash
• flask run
```

https://docs.docker.com/compose/gettingstarted/

Run the application

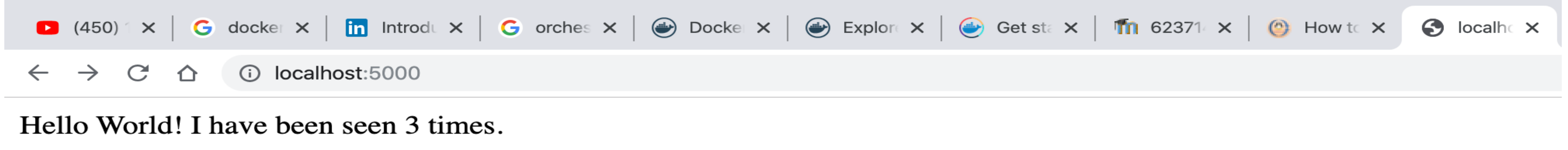


```
narges@narges-ThinkCentre-M910s:~/composetest$ docker exec -it erst /bin/bash
root@1c3b2e384aec:/# redis-server --port 6370
4621:C 09 May 23:36:23.756 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
4621:C 09 May 23:36:23.756 # Redis version=4.0.9, bits=64, commit=00000000, modified=0, pid=4621, just started
4621:C 09 May 23:36:23.756 # Configuration loaded

                                        Redis 4.0.9 (00000000/0) 64 bit

                                        Running in standalone mode
                                        Port: 6370
                                        PID: 4621

                                              http://redis.io
```



Hello World! I have been seen 3 times.

Note

sudo lsof -t -i:5000

- ❖**sudo** - command to ask admin privilege(user id and password).
- ❖**lsof** - list of files(Also used for to list related processes)
- ❖**-t** - show only process ID
- ❖**-i** - show only internet connections related process
- ❖**:5000** - show only processes in this port number

Be Careful which process you delete:

- ❖kill $(sudo lsof -t -i:5000)