SPRAWOZDANIE

PROGRAMOWANIE W CHMURZE OBLICZENIOWEJ

IMIĘ I NAZWISKO: Piotr Czajka
NUMER LABORATORIUM 7
GRUPA: 7.1.2
Data wykonywania ćwiczenia: 07.12.2018

Spis treści

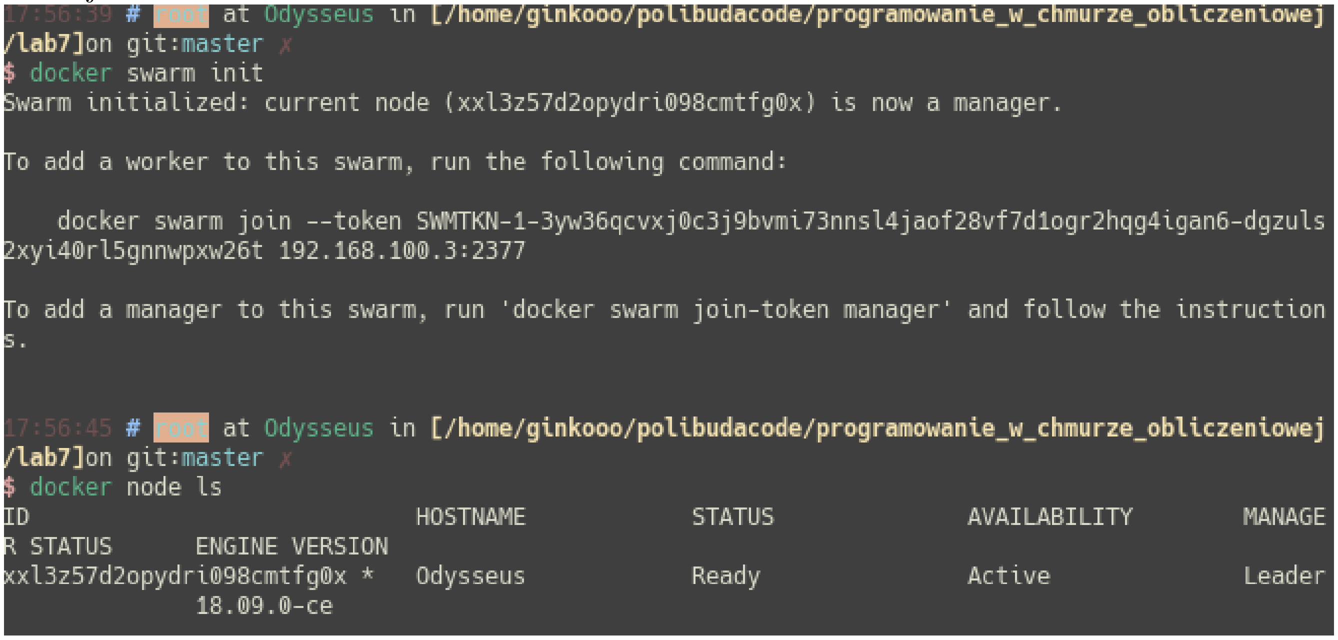
1 Cel laboratorium

Celem laboratorium było zapoznanie się z tworzeniem klastrów Swarm.

2 Przebieg ćwiczenia

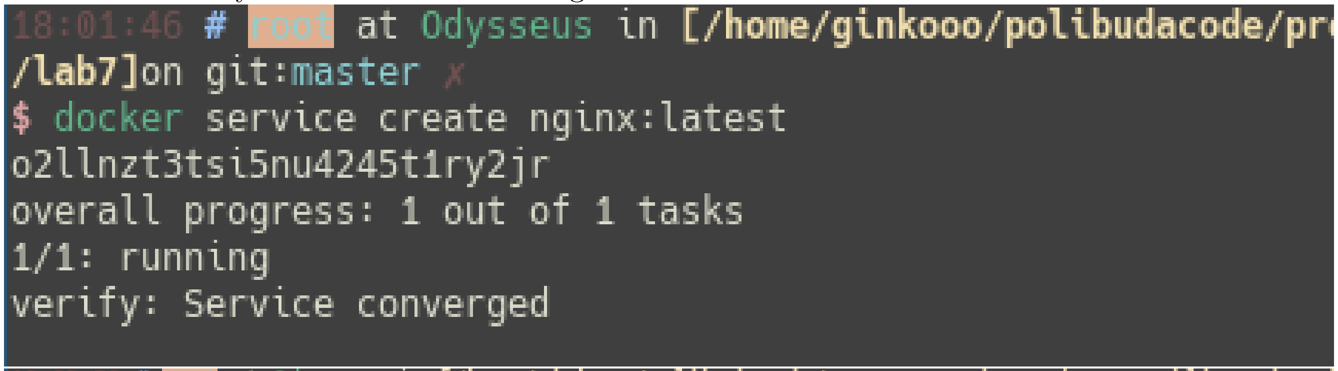
2.1 Zadanie pierwsze

Zainicjowano klaster swarm:

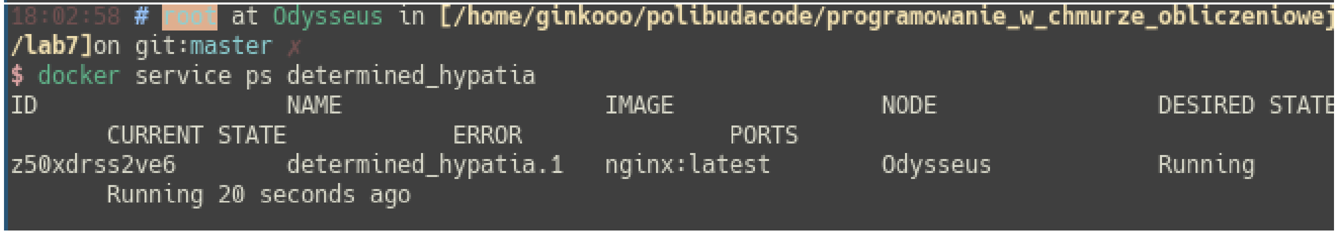


```
17:56:39 # root at Odysseus in [/home/ginkooo/polibudacode/programowanie_w_chmurze_obliczeniowej
/lab7]on git:master ✗
$ docker swarm init
Swarm initialized: current node (xxl3z57d2opydri098cmtfg0x) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3yw36qcvxj0c3j9bvmi73nnsl4jaof28vf7d1ogr2hqg4igan6-dgzuls
2xyi40rl5gnnwpxw26t 192.168.100.3:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instruction
s.


17:56:45 # root at Odysseus in [/home/ginkooo/polibudacode/programowanie_w_chmurze_obliczeniowej
/lab7]on git:master ✗
$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGE
R STATUS      ENGINE VERSION
xxl3z57d2opydri098cmtfg0x *   Odysseus            Ready               Active              Leader
              18.09.0-ce
```

Stworzono i zweryfikowano działanie usługi w klastrze:

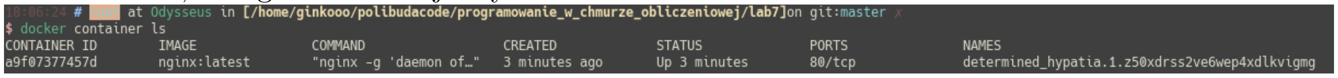


```
18:01:46 # root at Odysseus in [/home/ginkooo/polibudacode/pr
/lab7]on git:master ✗
$ docker service create nginx:latest
o2llnzt3tsi5nu4245t1ry2jr
overall progress: 1 out of 1 tasks
1/1: running
verify: Service converged
```



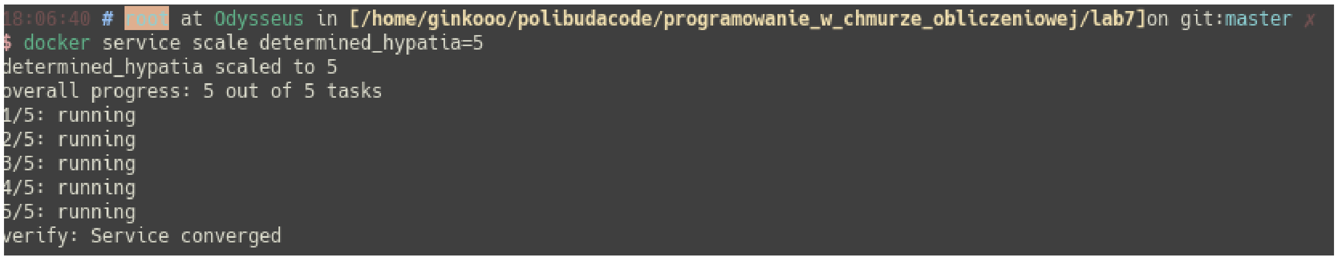
```
18:02:58 # root at Odysseus in [/home/ginkooo/polibudacode/programowanie_w_chmurze_obliczeniowej
/lab7]on git:master ✗
$ docker service ps determined_hypatia
ID                  NAME                   IMAGE               NODE                DESIRED STATE
       CURRENT STATE            ERROR               PORTS
z50xdrss2ve6        determined_hypatia.1   nginx:latest        Odysseus            Running
       Running 20 seconds ago
```

Jak widać tu, usługa działa na jednym kontenerze:



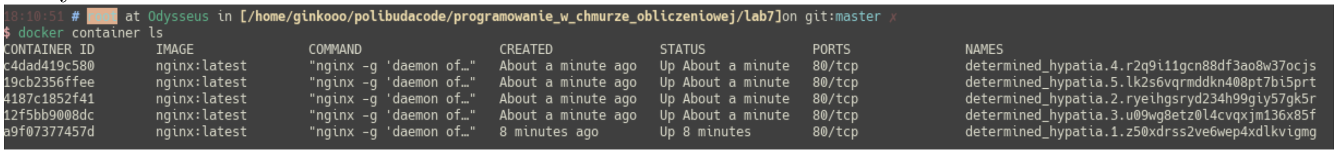
```
18:06:24 # root at Odysseus in [/home/ginkooo/polibudacode/programowanie_w_chmurze_obliczeniowej/lab7]on git:master ✗
$ docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
a9f07377457d        nginx:latest        "nginx -g 'daemon of…"   3 minutes ago       Up 3 minutes        80/tcp              determined_hypatia.1.z50xdrss2ve6wep4xdlkvigmg
```

Tu skaluje usługę na 5 kontenerów:



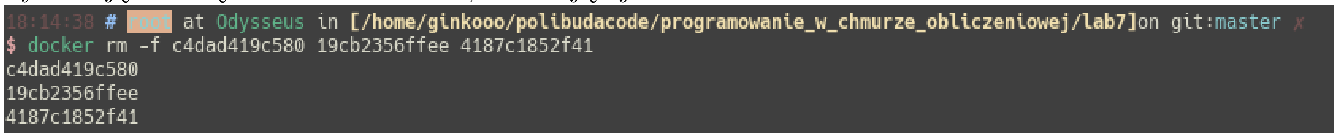
```
8:06:40 # root at Odysseus in [/home/ginkooo/polibudacode/programowanie_w_chmurze_obliczeniowej/lab7]on git:master ✗
$ docker service scale determined_hypatia=5
determined_hypatia scaled to 5
overall progress: 5 out of 5 tasks
1/5: running
2/5: running
3/5: running
4/5: running
5/5: running
verify: Service converged
```

Wszystko działa na 5 kontenerach:



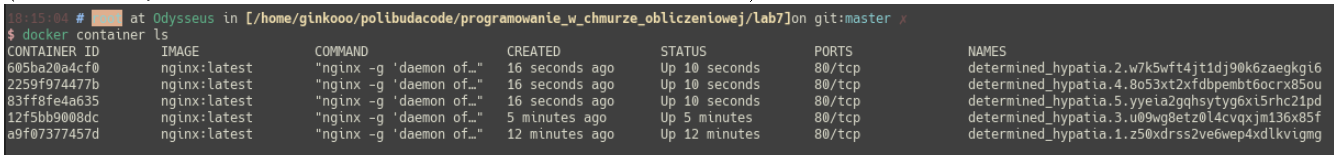
```
18:10:51 # root at Odysseus in [/home/ginkooo/polibudacode/programowanie_w_chmurze_obliczeniowej/lab7]on git:master ✗
$ docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS              PORTS               NAMES
c4dad419c580        nginx:latest        "nginx -g 'daemon of…"   About a minute ago   Up About a minute   80/tcp              determined_hypatia.4.r2q9i11gcn88df3ao8w37ocjs
19cb2356ffee        nginx:latest        "nginx -g 'daemon of…"   About a minute ago   Up About a minute   80/tcp              determined_hypatia.5.lk2s6vqrmddkn408pt7bi5prt
4187c1852f41        nginx:latest        "nginx -g 'daemon of…"   About a minute ago   Up About a minute   80/tcp              determined_hypatia.2.ryeihgsryd234h99giy57gk5r
12f5bb9008dc        nginx:latest        "nginx -g 'daemon of…"   About a minute ago   Up About a minute   80/tcp              determined_hypatia.3.u09wg8etz0l4cvqxjm136x85f
a9f07377457d        nginx:latest        "nginx -g 'daemon of…"   8 minutes ago        Up 8 minutes        80/tcp              determined_hypatia.1.z50xdrss2ve6wep4xdlkvigmg
```

Symuluję awarię 3 z 5 kontenerów, usuwając je:



```
18:14:38 # root at Odysseus in [/home/ginkooo/polibudacode/programowanie_w_chmurze_obliczeniowej/lab7]on git:master ✗
$ docker rm -f c4dad419c580 19cb2356ffee 4187c1852f41
c4dad419c580
19cb2356ffee
4187c1852f41
```

Tutaj widzimy, że Docker sam sobie stworzył nowe kontenery, w miejsce tych uszkodzonych (Można się zorientować po innym ID i któtkim uptime):



```
18:15:04 # root at Odysseus in [/home/ginkooo/polibudacode/programowanie_w_chmurze_obliczeniowej/lab7]on git:master ✗
$ docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
605ba20a4cf0        nginx:latest        "nginx -g 'daemon of…"   16 seconds ago      Up 10 seconds       80/tcp              determined_hypatia.2.w7k5wft4jt1dj90k6zaegkgi6
2259f974477b        nginx:latest        "nginx -g 'daemon of…"   16 seconds ago      Up 10 seconds       80/tcp              determined_hypatia.4.8o53xt2xfdbpembt6ocrx85ou
83ff8fe4a635        nginx:latest        "nginx -g 'daemon of…"   16 seconds ago      Up 10 seconds       80/tcp              determined_hypatia.5.yyeia2gqhsytyg6xi5rhc21pd
12f5bb9008dc        nginx:latest        "nginx -g 'daemon of…"   5 minutes ago       Up 5 minutes        80/tcp              determined_hypatia.3.u09wg8etz0l4cvqxjm136x85f
a9f07377457d        nginx:latest        "nginx -g 'daemon of…"   12 minutes ago      Up 12 minutes       80/tcp              determined_hypatia.1.z50xdrss2ve6wep4xdlkvigmg
```

2.2 Zadanie drugie

Stworzono następujące pliki w katalogu 'friendlyhello':

app.py

```python
import os
import socket

from flask import Flask
from redis import Redis, RedisError


redis = Redis(host='redis', db=0, socket_connect_timeout=2, socket_timeout=2)

app = Flask(__name__)

@app.route('/')
def hello():
    try:
        visits = redis.incr('counter')
    except RedisError:
        visits = '<i>cannot connect to Redis, counter disabled</i>'

    html = '''
    <h3>Hello {name}!</h3>
    <b>Hostname:</b> {hostname}<br />
    <b>Visits:</b> {visits}
    '''
    return html.format(name=os.getenv('NAME', 'world'),
        hostname=socket.gethostname(),
        visits=visits)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80)
```

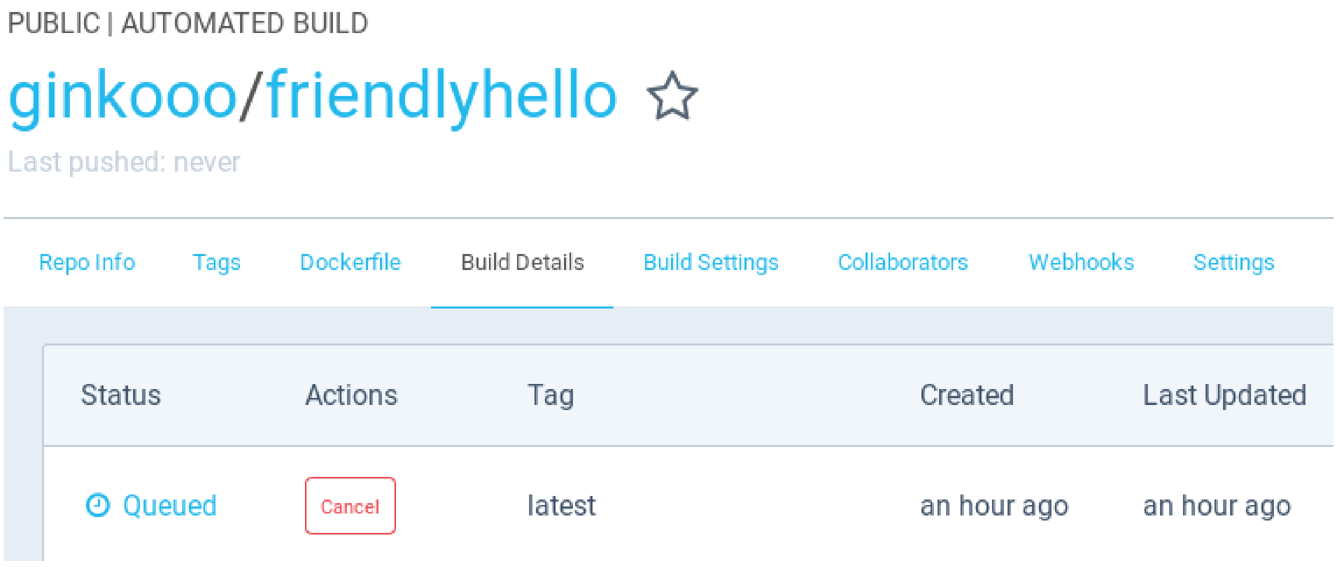
requirements.txt

```
Flask==1.0.2
redis==3.0.1
```

Dockerfile

```
FROM python:3.7
WORKDIR /app
COPY . /app
RUN pip install --trusted-host pypi.python.org -r requirements.txt
EXPOSE 80
ENV NAME Word
CMD ["python", "app.py"]
```

Połączono DockerHub z GitHub, a obraz jest zakolejkowany do budowania:

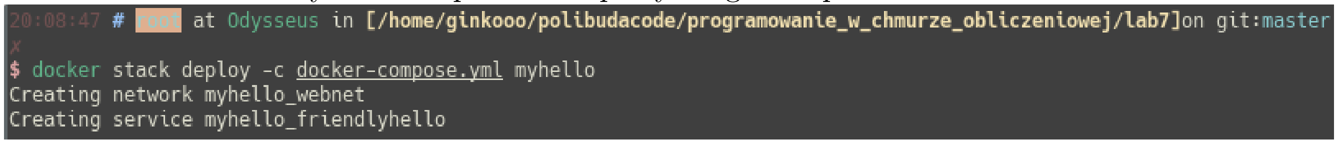


Status	Actions	Tag	Created	Last Updated
Queued	Cancel	latest	an hour ago	an hour ago

Link do githuba: github.com/ginkooo/friendlyhello
Napisano taki oto docker-compose.yml:

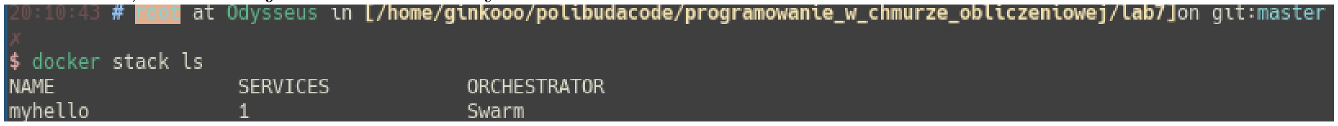
```
version: '3'
services:
    friendlyhello:
        image: ginkooo/friendlyhello
        deploy:
            replicas: 5
            resources:
                limits:
                    cpus: '0.1'
                    memory: 50M
            restart_policy:
                condition: on-failure
        ports:
            - '4000:80'
        networks:
            - webnet
networks:
    webnet:
```

Uruchomiono stack 'myhello' na podstawie powyższego compose:



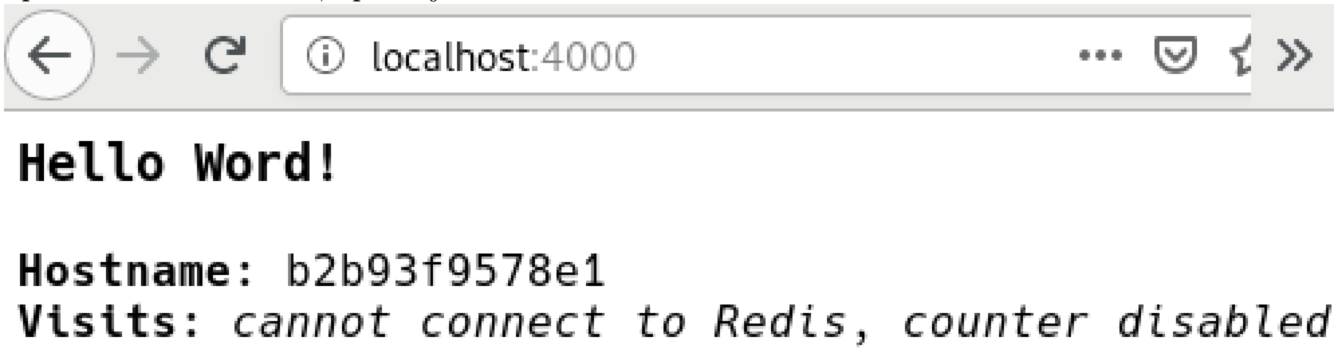
```
20:08:47 # root at Odysseus in [/home/ginkooo/polibudacode/programowanie_w_chmurze_obliczeniowej/lab7]on git:master
✗
$ docker stack deploy -c docker-compose.yml myhello
Creating network myhello_webnet
Creating service myhello_friendlyhello
```

Tu widać, że stack jest uruchomiony:



```
20:10:43 # root at Odysseus in [/home/ginkooo/polibudacode/programowanie_w_chmurze_obliczeniowej/lab7]on git:master
✗
$ docker stack ls
NAME                SERVICES            ORCHESTRATOR
myhello             1                   Swarm
```

Sprawdzono w firefoxie, aplikacja działa:

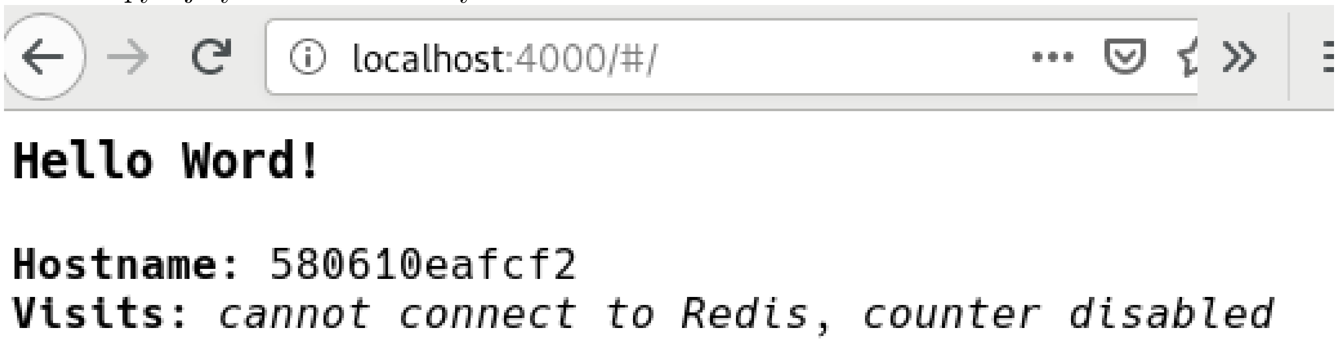


localhost:4000

```
Hello Word!

Hostname: b2b93f9578e1
Visits: cannot connect to Redis, counter disabled
```

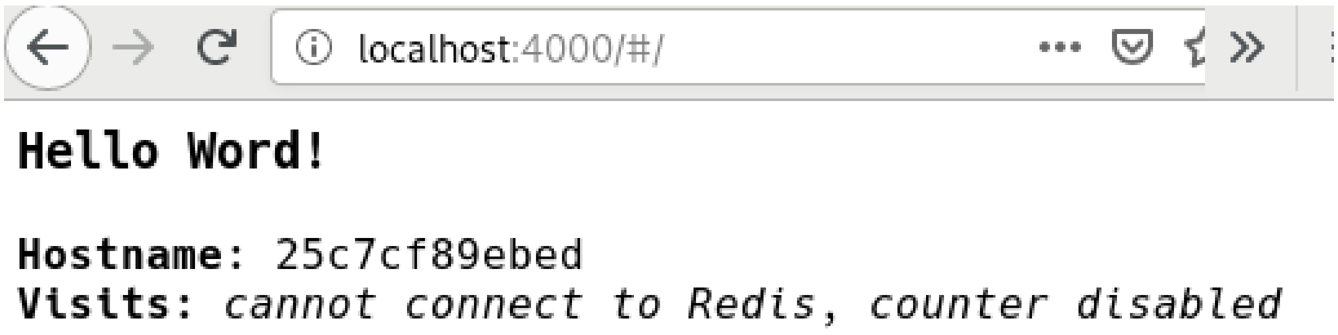
Tą komendą możemy przeskalować usługę do 7 replik

Teraz odpytajmy serwer kilka razy:



localhost:4000/#/

```
Hello Word!

Hostname: 580610eafcf2
Visits: cannot connect to Redis, counter disabled
```



localhost:4000/#/

```
Hello Word!

Hostname: 25c7cf89ebed
Visits: cannot connect to Redis, counter disabled
```

Rządania są dynamicznie oddelegowywane do różnych kontenerów je wykonujących. Efektem tego jest zmiana wartości w linijce Hostname.