Lab 10 Report - Daria Vaskovskaya

Chosen task: intelligent load balancing.

Configuration

Endpoint apps will use `flask` and serve a single number over HTTP.

App sources:

```
import sys
from flask import Flask


app = Flask("name")


@app.route("/")
def index():
    return str(sys.argv[1:])


app.run(host="0.0.0.0", port=5000)
```

We will run them using docker.

`Dockerfile`, with `[#]` meaning app number (1 to 3):

```
FROM python:3.6-alpine

WORKDIR /app
CMD python3 app.py [#]
RUN pip3 install flask

COPY ./app.py .
```

Load balancing

For load balancing we will use `traefik`.

It uses two configuration sources - static and dynamic. Static defines startup parameters and ingress ports, and cannot be changed without restarting the proxy. On the other hand, dynamic configuration can be hot-reloaded, and defines the backend services which are behind this proxy.

We will use dynamic configuration from file, and static configuration from command line. We will also use `docker-compose` to run everything.

`dyn-traefik.yml`:

```
http:
  routers:
    to-app:
      entryPoints:
        - web
      rule: Path(`/`)
      service: app

  services:
    app:
      loadBalancer:
        servers:
          - url: "http://app1:5000/"
          - url: "http://app2:5000/"
          - url: "http://app3:5000/"
        healthCheck:
          path: "/"
          interval: "2s"
          timeout: "1s"
```

`docker-compose.yml`:

```
version: '3'

services:
  app1:
    build:
      context: .
      dockerfile: Dockerfile.app1
  app2:
```

```
    build:
      context: .
      dockerfile: Dockerfile.app2
  app3:
    build:
      context: .
      dockerfile: Dockerfile.app3
  traefik:
    image: traefik:2.0.2
    ports:
      - 8080:80
    volumes:
      - ./traefik.yml:/etc/traefik/dyn-traefik.yml
    command:
      - --providers.file.filename=/etc/traefik/dyn-traefik.yml
      - --entryPoints.web.address=:80
```

Testing

Let's fire this up:

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

And check the balancing:

```
$ curl http://localhost:8080/
['1']
$ curl http://localhost:8080/
['2']
$ curl http://localhost:8080/
['3']
```

As we can see, it works. Let's try disabling one node.

```
$ curl http://localhost:8080/
['1']
$ curl http://localhost:8080/
['3']
$ curl http://localhost:8080/
['1']
```

As we can see, this also works.