SNA Lab Report

Sergey Bogdanik

January 17, 2020

1 Nginx

Nginx image configured with this config file:

```
http {
    server {
        listen 80;
        location / {
            proxy_pass http://app:5000/;
        }
    }
}
events {}
```

And run with following configuration in compose file:

```
  nginx:
    image: nginx
    ports:
      - 80:80
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf
    networks:
      - internal
      - external
```

It runs the official nginx image as a proxy server which forwards http requests from port 80 of the server in the external network to port 5000 of the

app container in the internal network.

2 App

Flask application with following source code responsible for sending log messages:

```
def send(level):
        logger = sender.FluentSender('app', host='fluentd', port=24224)
        result = logger.emit('flask.' + level, {'level': level,\
                'from': 'UserA', 'to': 'UserB'})
        if not result:
                error = logger.last_error
                logger.clear_last_error()
        else:
                error = 'OK'
        logger.close()
        return error
```

The sender send logs to another container named fluentd to via port 24224. A log is sent only when the special page is opened. Here is the template of possible pages:

```
<!DOCTYPE html>
<html>
<body>
  <ul>
    <li><p><a href="{{ url_for('info') }}">
      Send <b>info</b> log message</a></p></li>
    <li><p><a href="{{ url_for('debug') }}">
      Send <b>debug</b> log message</a></p></li>
    <li><p><a href="{{ url_for('warning') }}">
      Send <b>warning</b> log message</a></p></li>
    <li><p><a href="{{ url_for('error') }}">
      Send <b>error</b> log message</a></p></li>
  </ul>
</body>
```

```
</html>
```

The application has following imports:

```
from flask import Flask, render_template
from fluent import sender
```

Therefore it should install following requirements:

```
Flask
fluent-logger
```

Finaly the application is run in the official python3 docker image with the following Dockerfile:

```
FROM python:3

WORKDIR /usr/src/app

COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 5000

CMD [ "python", "./app.py" ]
```

Port 5000 is exposed because it's the default port of Flask applications. The application is run with following configuration in compose file:

```
  app:
    build: ./app
    ports:
      - 5000:5000
    networks:
      - internal
```

3 Fluentd

The fluentd image with following config:

```
<source>
        @type forward
        port 24224
        bind 0.0.0.0
</source>

<match **>
        @type mongo
        database admin
        host mongodb
        port 27017
        collection logs
        user root
        password pass
        <buffer>
                flush_interval 10s
        </buffer>
        <inject>
                time_key time
        </inject>
</match>
```

It has one source of logs. Port 24224 listens for connections from any IP. Also it sends all logs (even fluentd logs) to one destination. Mongo database located on host mongodb in another docker container. It sends logs to defualt database admin through default port 27017. It creates collection named logs. And uses user root with password pass to authenticate. It flushes buffer every 10 seconds and adds time field automatically.

The Dockerfile used for this image is following:

```
FROM fluent/fluentd

RUN apk add --no-cache bash make gcc libc-dev ruby-dev \
    && gem install fluent-plugin-mongo \
    && apk del make gcc libc-dev ruby-dev \
    && rm -rf /var/cache/apk/* \
    && rm -rf /tmp/* /var/tmp/* /usr/lib/ruby/gems/*/cache/*.gem
```

It takes official fluentd image and installs mongo plugin. The image is run with following configuration in compose file:

```
  fluentd:
    build: ./fluent
    ports:
      - 24224:24224
    volumes:
      - './fluent/fluent.conf:/fluentd/etc/fluent.conf'
    networks:
      - internal
```

4 Mongo

Mongodb is used instead of influxdb because fluent-logger library doesn't work well with influxdb output plugin. Sometimes the library implicitly creates fileds in a dictionary with empty string values and influxdb plugin can't write them into database. The issue (https://github.com/fangli/fluent-plugin-influxdb/issues/96) is already opened.
It's a default mongodb image downloaded from docker-hub and run without any changes in docker-compose. The following configuration in compose file is used:

```
mongodb:
  image: mongo
  ports:
    - 27017:27017
  environment:
    MONGO_INITDB_ROOT_USERNAME: root
    MONGO_INITDB_ROOT_PASSWORD: pass
  networks:
    - internal
```

It runs the image publishing default 27017 port and setting 2 environment variables used for authentication. Root user named root and it's password pass.