Simon Gomez - Microservices

Monitoring your applications

Monitor Logs

step 1 : install grafana

we are going to create a dockercompose file for these monitoring solution

add grafana in your dockercompose file :

```
grafana:
    image: grafana/grafana
    container_name: grafana
    ports:
      - 3000:3000
    restart: unless-stopped
    networks:
      - monitor-net
```

check that grafana is working by connecting on localhost:3000, default password should be admin/admin

step 2 : install loki

add loki to your docker compose

```
  loki:
    image: grafana/loki
    container_name: loki
    ports:
      - 3100:3100
    restart: unless-stopped
    networks:
      - monitor-net
```

step 3 : add some logs in your app

add winston and windston-loki package via npm install

Setup the winston configuration

```
const loki_uri = process.env.LOKI || "http://127.0.0.1:3100";


const { createLogger, transports } = require("winston");
const LokiTransport = require("winston-loki");
const options = {
  transports: [
    new LokiTransport({
      host: loki_uri
    })
  ]
};
```

Setup some logs in your app

```
  logger.info({ message: 'URL '+req.url , labels: { 'url': req.url, 'user':username } })
```

Try it out

add loki datasource in your grafana go to the explore panel try to find your logs

Add some Metrics

step 1 : install prometheus

add prometheus to your dockercompose

add prometheus in your dockercompose file :

```
  prometheus:
    image: prom/prometheus
    container_name: prometheus
    volumes:
      - ./prometheus/:/etc/prometheus/
    restart: unless-stopped
    command:
      - '--config.file=/etc/prometheus/prometheus.yml'
    expose:
      - 9090
    ports:
      - 9090:9090
    networks:
      - monitor-net
```

configure prometheus

create a file prometheus.yml in a prometheus directory

```
global:
  scrape_interval: 1m

scrape_configs:
  - job_name: "prometheus"
    scrape_interval: 1m
    static_configs:
    - targets: ["localhost:9090"]
```

- run your dockercompose
- connect to your grafana
- add the prometheus instance as a datasource (You should see “Data source is working” in your grafana interface

step 2: add the node exporter to monitor your system

add a node exporter instance in your dockercompose file

```
   node-exporter:
    image: prom/node-exporter:latest
    container_name: node-exporter
    restart: unless-stopped
    volumes:
      - /proc:/host/proc:ro
      - /sys:/host/sys:ro
      - /:/rootfs:ro
    command:
      - '--path.procfs=/host/proc'
      - '--path.rootfs=/rootfs'
      - '--path.sysfs=/host/sys'
      - '--collector.filesystem.mount-points-exclude=^/(sys|proc|dev|host|etc)($$|/)'
    expose:
      - 9100
```

```
    ports:
      - 9100:9100
    networks:
      - monitor-net
```

Connect to http://localhost:9100/metrics and have a look to all data gathered

Add the node exporter to your prometheus configuration

```
     - job_name: "node-exporter"
    static_configs:
    - targets: ["node-exporter:9100"]
```

Restart your docker compose try graphing the prometheus collect :

- new dashboard
- new panel
- use your prometheus datasource
- graph : promhttp_metric_handler_requests_total
- stop the node exporter
- look a the graph

step 3 : add metrics in your application

add an API /metrics in your motus app This API will return prometheus format metrics with the number of request done and the number of successful authentication Here is an example :

```
http_requests_total XX
login_total YY
```

add the collect in your prometheus configuration

```
- job_name: "mynode"
  scrape_interval: 10s
  static_configs:
  - targets: ["mynodeapp:4000"]
```

graph all these information

Published with GitHub Pages