Exploring the ELK Stack for Monitoring APIs with Express

Create a File in Express

```
import express, { Request, Response } from 'express';
import { createLog } from './utils/commonUtils';

const app = express();
const port = 8090;

app.get('/', (req: Request, res: Response) => {
  res.send('Hello Bayu Widia Santoso!');
});

app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);

  createLog("info","info",`Server is running on port http://localhost:${port}`,200,"",0,null, );

});
```

Then, add these two methods to call it, distinguishing between "info" and "error".

```
app.get('/info', (req: Request, res: Response) => {
  createLog("success","success","Log has been written! Check the log file.",200,"",0,null, );

  res.json({
    status: 'success',
    message: 'Log has been written! Check the log file.'
  });
});
```

```
app.get('/error', (req: Request, res: Response) => {
  const errorMsg = 'Something went wrong!';

  createLog("error","error",`Error occurred at /error endpoint: ${errorMsg}`,200,"",0,null, );

  res.json({
    status: 'error',
    message: errorMsg
  });
});
```

```typescript
import winston from 'winston';
import DailyRotateFile from 'winston-daily-rotate-file';
import path from 'path';

// Define a custom log format
const customFormat = winston.format.printf(({ level, timestamp, message, title, code, endpoint, linenumber, datas })
=> {
    return JSON.stringify({
      'log.level': level,
      '@timestamp': timestamp,
      message,
      title,
      code,
      endpoint,
      linenumber,
      datas,
    });
});

const logger = winston.createLogger({
    format: winston.format.combine(
        winston.format.timestamp({ format: 'YYYY-MM-DDTHH:mm:ssZ' }),
        customFormat
    ),
    transports: [
        new winston.transports.Console(),
        new DailyRotateFile({
            filename: path.join(__dirname, '../logs', 'nodejs-typescript-elk-%DATE%.log'),
            format: winston.format.combine(winston.format.uncolorize()),
            datePattern: 'YYYY-MM-DD',
            zippedArchive: true,            // Mengarsipkan log yang telah dirotasi
            maxSize: '20m',                 // Ukuran maksimum untuk setiap file log
            maxFiles: '14d',                // Simpan log selama 14 hari
            auditFile: path.resolve(__dirname, '../logs/audit', '.audit.json')  // Atur lokasi file audit
        }),
    ],
});

export default logger;
```

Create a File name logger.ts

```typescript
import logger from '../logging/logger';

export function createLog(logFlagging:string, title:string, message:string, code:number, endpoint:string,
linenumber:number, datas:any) {

    if (logFlagging === "success") {
        logger.info({
            message: message,
            title: title,
            code: code,
            endpoint: endpoint,
            linenumber: linenumber,
            datas: datas,
        });
    } else if (logFlagging === "info") {
        logger.info({
            message: message,
            title: title,
            code: code,
            endpoint: endpoint,
            linenumber: linenumber,
            datas: datas,
        });
    } else if (logFlagging === "error") {
        logger.error({
            message: message,
            title: title,
            code: code,
            endpoint: endpoint,
            linenumber: linenumber,
            datas: datas,
        });
    } else {
        logger.error('logFlagging No match found');
    }
}
```

Create a File name commonUtils.ts

Once that's done, we will prepare some ELK stack configurations.

We will create several configuration files, including:

Elasticsearch
Filebeat
Kibana
Logstash

Elasticsearch

```
//This is the Filebeat file.

filebeat.inputs:
  - type: log
    enabled: true
    paths:
      - /src/logs/*.log

output.logstash:
  hosts: ["logstash:5044"]
```

Filebeat

```
//This is the Elasticsearch file.

network.host: 0.0.0.0
discovery.type: single-node
xpack.security.enabled: false
```

kibana

```
//This is the Kibana file.

server.host: "0.0.0.0"
elasticsearch.hosts: [ "http://elasticsearch:9200" ]
```

Logstash

```
//This is the logstash file.

input {
  beats {
    port => 5044
  }
}

filter {
  json {
    source => "message"  # Mengonversi message menjadi format JSON
  }
}

output {
  elasticsearch {
    hosts => ["http://elasticsearch:9200"]  # Alamat Elasticsearch
    index => "nodejs-typescript-elk-%{+yyyy.MM.dd}"  # Index pattern
  }
}
```

Once it's done, we will prepare the ELK stack to run using Docker.

Create a Dockerfile for the application

```
# Menggunakan image Node.js 18 Alpine
FROM node:18-alpine

# Set working directory di dalam container
WORKDIR /usr/src/app

# Menyalin file package.json dan package-lock.json
COPY package*.json ./

# Install dependensi aplikasi
RUN npm install

# Menyalin seluruh file aplikasi ke dalam container
COPY . .

# Build aplikasi TypeScript ke JavaScript
RUN npm run build

# Menjalankan aplikasi dari dist
CMD ["node", "dist/index.js"]
```

Prepare the dockerfile.elasticsearch file, which will later be called in the docker-compose file.

```
# Elasticsearch Dockerfile
FROM elasticsearch:8.8.0

# Beralih ke pengguna root untuk menjalankan perintah izin
USER root

# Salin dan atur izin konfigurasi Elasticsearch
COPY ./config/elasticsearch.yml /usr/share/elasticsearch/config/elasticsearch.yml
RUN chown elasticsearch:elasticsearch /usr/share/elasticsearch/config/elasticsearch.yml && \
    chmod go-w /usr/share/elasticsearch/config/elasticsearch.yml

# Kembalikan ke pengguna elasticsearch untuk keamanan
USER elasticsearch

# Ekspose port
EXPOSE 9200 9300
```

Prepare the dockerfile.filebeat file, which will later be called in the docker-compose file.

```
# Filebeat Dockerfile
FROM docker.elastic.co/beats/filebeat:8.8.0

# Beralih ke root untuk mengubah izin
USER root

# Salin konfigurasi Filebeat ke dalam container
COPY ./config/filebeat.yml /usr/share/filebeat/filebeat.yml

# Atur izin untuk file konfigurasi
RUN chmod go-w /usr/share/filebeat/filebeat.yml

# Jalankan Filebeat
CMD ["filebeat", "-e", "-c", "/usr/share/filebeat/filebeat.yml"]
```

Prepare the dockerfile.kibana file, which will later be called in the docker-compose file.

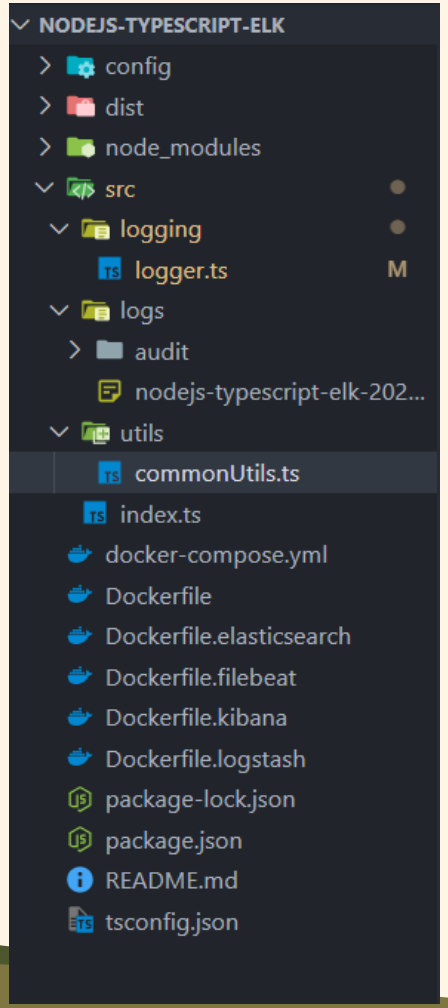
```
# Kibana Dockerfile
FROM kibana:8.8.0

# Salin konfigurasi Kibana (jika ada konfigurasi tambahan)
COPY ./config/kibana.yml /usr/share/kibana/config/kibana.yml

# Ekspose port
EXPOSE 5601
```

Prepare the dockerfile.logstach file, which will later be called in the docker-compose file.

```
# Logstash Dockerfile
FROM logstash:8.8.0

# Salin konfigurasi Logstash ke dalam container
COPY ./config/logstash.conf /usr/share/logstash/pipeline/logstash.conf

# Ekspose port
EXPOSE 5044
```

Set up a Docker Compose configuration.

```yaml
services:
  express-app:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "8050:8050"
    volumes:
      - ./src:/usr/src/app/src  # Mount folder src ke container
      - ./dist:/usr/src/app/dist  # Mount folder dist ke dalam container
      - ./src/log:/usr/src/app/log  # Mount folder log ke dalam container
    environment:
      - NODE_ENV=production

  elasticsearch:
    build:
      context: .
      dockerfile: Dockerfile.elasticsearch
    environment:
      - discovery.type=single-node
    ports:
      - "9200:9200"
    volumes:
      - esdata:/usr/share/elasticsearch/data

  logstash:
    build:
      context: .
      dockerfile: Dockerfile.logstash
    ports:
      - "5044:5044"
    depends_on:
      - elasticsearch

  kibana:
    build:
      context: .
      dockerfile: Dockerfile.kibana
    ports:
      - "5601:5601"
    depends_on:
      - elasticsearch

  filebeat:
    build:
      context: .
      dockerfile: Dockerfile.filebeat
    volumes:
      - ./src/logs:/src/logs:delegated  # Volume mount yang sama untuk file log
    depends_on:
      - logstash

volumes:
  esdata:
```



The folder structure that has been implemented is as follows.

Once completed, it will generate a log like this, which can be processed using the ELK stack.


```
{"log.level":"info","@timestamp":"2024-11-10T06:38:21+07:00","message":"Server is running on port
http://localhost:8090","title":"info","code":200,"endpoint":"","linenumber":0,"datas":null}
{"log.level":"info","@timestamp":"2024-11-10T06:42:33+07:00","message":"Server is running on port
http://localhost:8090","title":"info","code":200,"endpoint":"","linenumber":0,"datas":null}
{"log.level":"info","@timestamp":"2024-11-10T06:47:42+07:00","message":"Server is running on port
http://localhost:8090","title":"info","code":200,"endpoint":"","linenumber":0,"datas":null}
{"log.level":"info","@timestamp":"2024-11-10T06:47:53+07:00","message":"Log has been written! Check the log
file.","title":"success","code":200,"endpoint":"","linenumber":0,"datas":null}
{"log.level":"error","@timestamp":"2024-11-10T06:47:57+07:00","message":"Error occurred at /error endpoint: Something
went wrong!","title":"error","code":200,"endpoint":"","linenumber":0,"datas":null}
{"log.level":"info","@timestamp":"2024-11-10T06:54:25+07:00","message":"Server is running on port
http://localhost:8090","title":"info","code":200,"endpoint":"","linenumber":0,"datas":null}
```

Here are the results that have been read by Kibana logs.

SEE YOU …