Exploring the ELK Stack for Monitoring APIs with Golang

Create a File in Golang

```go
package main

import (
    "go-fiber-elk/logging"
    "log"
    "net/http"
    "time"

    "github.com/gofiber/fiber/v2"
)

func main() {
    app := fiber.New()

    app.Get("/", func(c *fiber.Ctx) error {

        logging.WriteToLogger(c, "Hello Bayu Widia Santoso", "success", http.StatusBadRequest, "SUCCESS", nil)
        return c.SendString("Hello, Bayu widia Santoso!")
    })

    // Jalankan aplikasi
    log.Fatal(app.Listen(":8080"))
}
```

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

```go
app.Get("/error", func(c *fiber.Ctx) error {
    data := logging.ResponseData{
        ResponseCode:        http.StatusBadRequest,
        ResponseDescription: "Validasi input failed",
        ResponseTime:        time.Now().Format("2006-01-02 15:04:05"),
        ResponseDatas:       "Format JSON incorrect",
    }

    logging.WriteToLogger(c, "Ooopss incorrect JSON", "incorrect", http.StatusBadRequest, "ERROR", data)

    return c.Status(fiber.StatusBadRequest).SendString("Log Error recorded successfully!")
})
```

```go
app.Get("/info", func(c *fiber.Ctx) error {
    data := logging.ResponseData{
        ResponseCode:        http.StatusOK,
        ResponseDescription: "JSON Success",
        ResponseTime:        time.Now().Format("2006-01-02 15:04:05"),
        ResponseDatas:       "Format JSON SUCCESS",
    }

    logging.WriteToLogger(c, "Request Success", "success", http.StatusOK, "SUCCESS", data)

    return c.SendString("Log Info recorded successfully!")
})
```

Create a File name logger.go

```go
// logger.go
package logging

import (
    "path/filepath"
    "time"

    "github.com/gofiber/fiber/v2"
    "github.com/sirupsen/logrus"
    "gopkg.in/natefinch/lumberjack.v2"
)

var logger = logrus.New()

func logFileName() string {
    //local
    return filepath.Join("logs", "go-fiber-elk-"+time.Now().Format("02-Jan-2006")+".log")
    //docker desktop
    // return "/app/logs/go-fiber-elk-" + time.Now().Format("02-Jan-2006") + ".log"
}

func init() {
    logger.SetFormatter(&logrus.JSONFormatter{})

    logger.SetOutput(&lumberjack.Logger{
        Filename:   logFileName(),
        MaxSize:    10,   // Maks ukuran file dalam MB
        MaxBackups: 7,    // Jumlah file backup log
        MaxAge:     30,   // Maks usia file log dalam hari
        Compress:   true, // Mengompres file log lama
    })
}

func LogInfo(logData LogEntity) {
    entry := logger.WithFields(logrus.Fields{
        "log.level":  logData.LogLevel,
        "@timestamp": logData.Timestamp,
        "message":    logData.Message,
        "title":      logData.Title,
        "code":       logData.Code,
        "endpoint":   logData.Endpoint,
        "linenumber": logData.LineNumber,
        "datas":      logData.Datas,
        "latency":    logData.Latency,
    })

    if logData.LogLevel == "ERROR" {
        entry.Error(logData.Message)
    } else {
        entry.Info(logData.Message)
    }
}

func WriteToLogger(c *fiber.Ctx, message string, title string, code int, logLevel string, data interface{}) {

    LogInfo(LogEntity{
        Message:   message,
        Title:     title,
        Code:      code,
        Endpoint:  c.Path(),
        LogLevel:  logLevel,
        Timestamp: time.Now().Format(time.RFC3339),
        Datas:     data,
    })
}
```

```go
package logging

type ResponseData struct {
    ResponseCode        int          `json:"responseCode"`
    ResponseDescription string       `json:"responseDescription"`
    ResponseTime        string       `json:"responseTime"`
    ResponseDatas       interface{}  `json:"responseDatas"`
}

type LogEntity struct {
    Message    string       `json:"message"`
    Title      string       `json:"title"`
    Code       int          `json:"code"`
    Endpoint   string       `json:"endpoint"`
    LineNumber int          `json:"linenumber"`
    LogLevel   string       `json:"log.level"`
    Timestamp  string       `json:"@timestamp"`
    Datas      interface{}  `json:"datas"`
    Latency    string       `json:"latency"`
}
```

Create a File name logEntity.go

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

We will create several configuration files, including:

Elasticsearch
Filebeat
Kibana
Logstash

Elasticsearch

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

Filebeat

```
# config/filebeat.yml
filebeat.inputs:
  - type: log
    enabled: true
    paths:
      - /logs/*.log

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

kibana

```
# config/kibana.yml
server.host: "0.0.0.0"
elasticsearch.hosts: [ "http://elasticsearch:9200" ]
```

Logstash

```
input {
  beats {
    port => 5044
  }
}

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

output {
  elasticsearch {
    hosts => ["http://elasticsearch:9200"]  # Alamat Elasticsearch
    index => "go-fiber-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

```
FROM golang:1.20-alpine

WORKDIR /app

COPY go.mod go.sum ./

RUN go mod download

COPY . .

RUN go build -o /app/main .

EXPOSE 8080

CMD ["/app/main"]
```

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.

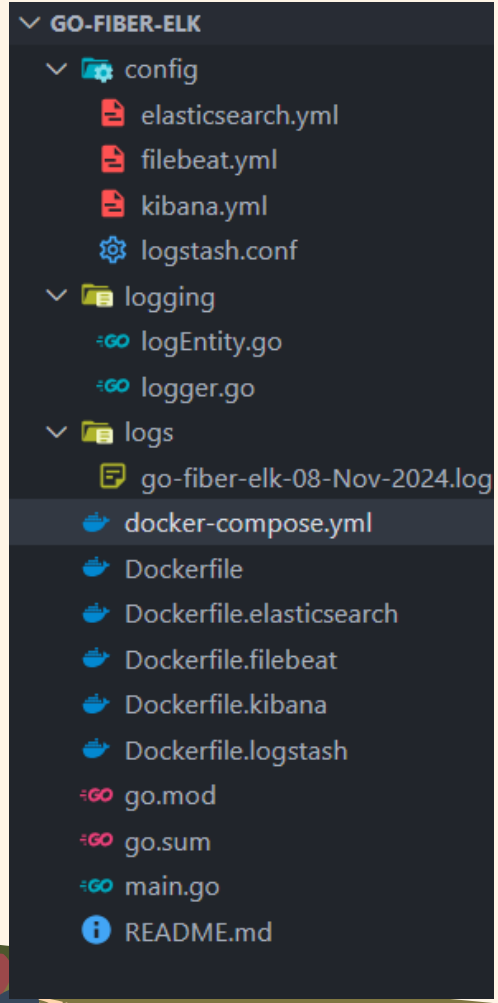
```
# Kibana Dockerfile
FROM kibana:8.8.0

# 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
version: '3'
services:
  app:
    build: .
    ports:
      - "8000:8000"
    environment:
      - ENV=development
    volumes:
      - ./logs:/app/logs:delegated  # Volume mount dengan opsi :delegated untuk sinkronisasi cepat

  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:
      - ./logs:/app/logs:delegated  # Volume mount yang sama untuk file log
    depends_on:
      - logstash

volumes:
  esdata:
```

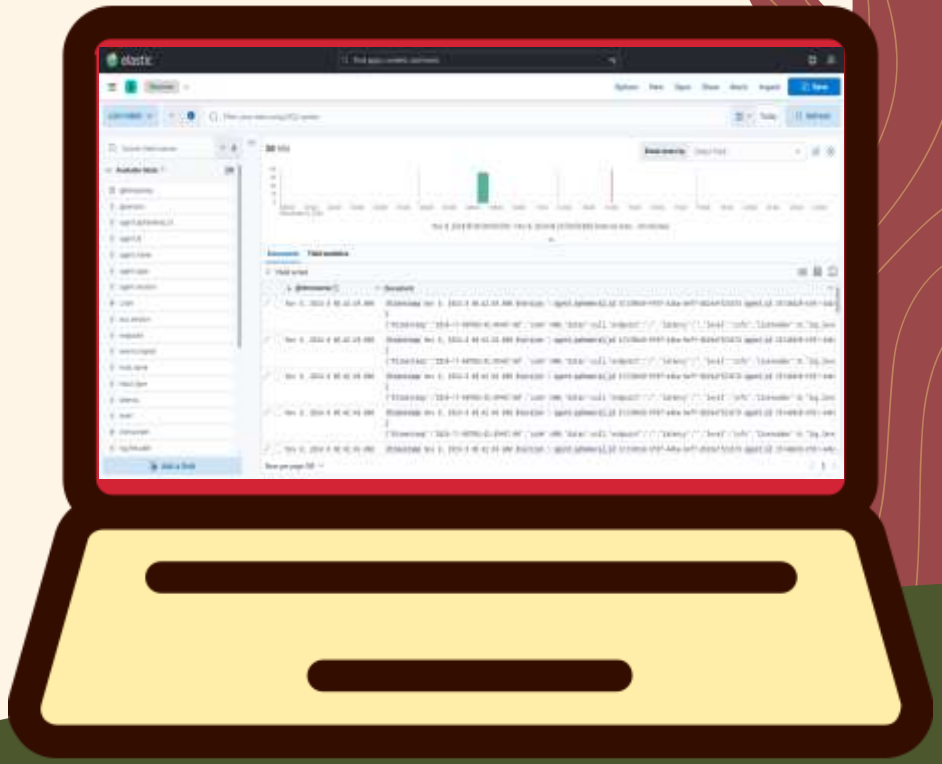


The folder structure that has been implemented is as follows.

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

```
{"@timestamp":"2024-11-
08T01:58:27Z","code":400,"datas":null,"endpoint":"/","latency":"","level":"info","linenumber":0,"log.level":"SUCCESS
","message":"Hello Bayu Widia Santoso","msg":"Hello Bayu Widia Santoso","time":"2024-11-
08T01:58:27Z","title":"success"}
{"@timestamp":"2024-11-
08T01:58:28Z","code":400,"datas":null,"endpoint":"/","latency":"","level":"info","linenumber":0,"log.level":"SUCCESS
","message":"Hello Bayu Widia Santoso","msg":"Hello Bayu Widia Santoso","time":"2024-11-
08T01:58:28Z","title":"success"}
{"@timestamp":"2024-11-
08T01:58:28Z","code":400,"datas":null,"endpoint":"/","latency":"","level":"info","linenumber":0,"log.level":"SUCCESS
","message":"Hello Bayu Widia Santoso","msg":"Hello Bayu Widia Santoso","time":"2024-11-
08T01:58:28Z","title":"success"}
{"@timestamp":"2024-11-
08T01:58:28Z","code":400,"datas":null,"endpoint":"/","latency":"","level":"info","linenumber":0,"log.level":"SUCCESS
","message":"Hello Bayu Widia Santoso","msg":"Hello Bayu Widia Santoso","time":"2024-11-
08T01:58:28Z","title":"success"}
{"@timestamp":"2024-11-
08T01:58:28Z","code":400,"datas":null,"endpoint":"/","latency":"","level":"info","linenumber":0,"log.level":"SUCCESS
","message":"Hello Bayu Widia Santoso","msg":"Hello Bayu Widia Santoso","time":"2024-11-
08T01:58:28Z","title":"success"}
```

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



SEE YOU …