UNIVERSITAS INDONESIA

MINI PROJECT 2

LAW - A

Disusun Oleh:

Muhammad Rizky Anditama **1706044080**

FAKULTAS ILMU KOMPUTER

UNIVERSITAS INDONESIA

2021

Prequisites

- Go
- Setup RabbitMQ to enable STOMP
  It can be done by using this command:
  ```
  rabbitmq-plugins enable rabbitmq_web_stomp
  ```

Penjelasan

Pada mini assignment ini, saya membuat dua web server, yang ada pada Server1 dan Server2. Server1 memiliki fungsi untuk mengembalikan html yang sesuai ketika user mengakses /upload. Kemudian, Server1 juga melayani upload untuk kemudian dikirim ke Server2. Server1 juga mengirim header X-ROUTING-KEY yang akan digunakan pada Server2 untuk mengirim pesan secara asinkronus ke client melalui RabbitMQ(routing key). Kemudian, server 1 juga merender html untuk progress ketika sudah mendapat balasan dari server 2.

Pada server 2, diinisiasi koneksi ke rabbitMQ dengan exchange “progress”. Setiap server 2 menerima request, server 2 membuat goroutines baru sehingga proses kompresi dan penyimpanan file dilakukan secara asynchronous(background process). Saya menggunakan library progress yang dibuat oleh akun github machinebox. Library ini akan digunakan untuk men-track progress dari proses IO yang terjadi, yaitu proses kompresi. Kemudian, buat goroutine baru lagi di dalam goroutine ini yang digunakan untuk secara asynchronous mengirimkan update terkait progress kompresi. Di akhir handler, terdapat pengiriman link file menuju exchange rabbitmq yang sebelumnya.

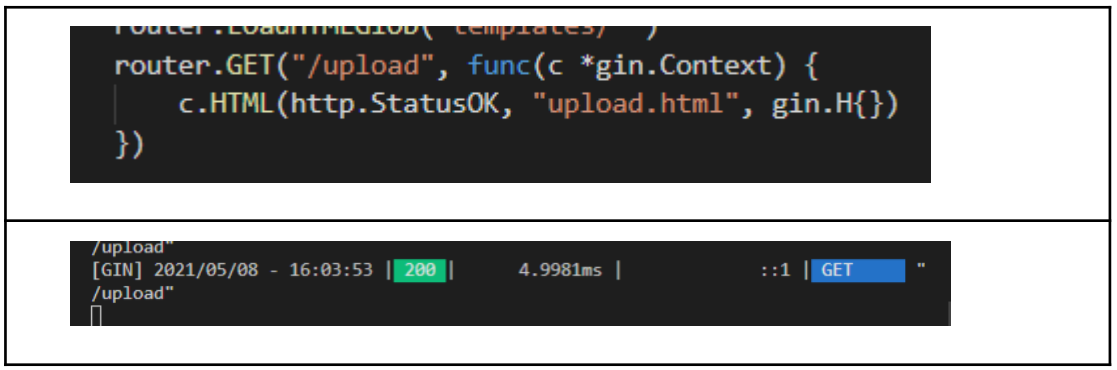
Pada file html upload.html, ini hanya berfungsi untuk menerima input file dan mengirimkannya ke server 1.

Pada file html progress.html, ketika pertama kali di load, client akan membuat koneksi ke stomp berdasarkan routing key yang diberikan oleh server 1. Client akan berkomunikasi dengan server 2 menggunakan koneksi rabbitmq ini. Client setiap kali menerima pesan dari server 2 akan melakukan aktivitas sesuai status yang diterima, berikut adalah deskripsinya:

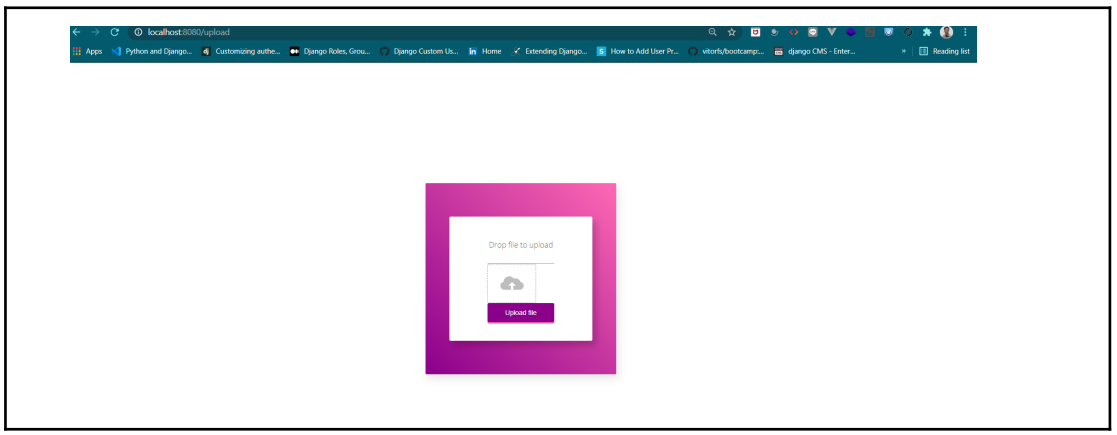
- **status: “In Progress”** → Mengganti atribut element progress sehingga tampilannya akan berubah
- **status “Saving file”** → Mengganti atribut element progress menjadi 100(selesai) sehingga tampilan progress barnya menjadi full
- **status “File saved”** → Mengirim pesan berupa link unduh file yang telah dikompres, kemudian men set href untuk button pada html sehingga ketika ditekan akan mengunduh file tersebut

Screenshots

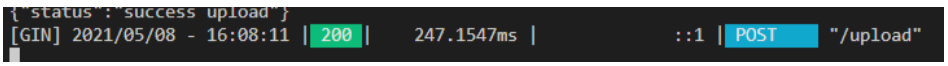
1. Laptop Mahasiswa mengirimkan request request halaman upload file dan diterima oleh Server 1



2. Server 1 menghasilkan halaman HTML (halaman form upload file) dan halaman HTML tersebut dikembalikan ke Laptop Mahasiswa



3. Laptop Mahasiswa mengirimkan file yang dipilih pada browser untuk diupload dan diterima oleh Server 1 (tanpa menyimpan permanen di Server 1)




```go
router.POST("/upload", func(c *gin.Context) {
    // single file
    fileInput, err := c.FormFile("file")
    if err != nil {
        c.JSON(http.StatusUnprocessableEntity, gin.H{
            "status": http.StatusUnprocessableEntity,
            "error":  "Unable to parse requeest",
        })
        return
    }

    // Buffer for temporary saving the ifle
    buf := new(bytes.Buffer)
    writer := multipart.NewWriter(buf)

    // Create multipart form file from file input
    part, err := writer.CreateFormFile("file", fileInput.Filename)
    if err != nil {
        fmt.Println(err)
    }
```

```go
// Open file IO
file, err := fileInput.Open()
if err != nil {
    fmt.Println(err)
}

// Copy content of file to multi part
if _, err = io.Copy(part, file); err != nil {
    fmt.Println(err)
}

writer.Close()
```

4. Sesaat file sudah diterima komplit pada Server 1, maka file tersebut dikirimkan ke Server 2. Pada saat Server 1 melakukan pemanggilan ke Server 2, pada header bernama “X-ROUTING-KEY” diisi sebuah UNIQ ID yang digenerate secara Random oleh Server 1.

```go
// Initiate http client and add body from previous buf
client := &http.Client{}
req, err := http.NewRequest("POST", "http://localhost:8081/compress", buf)

// Add request header
randomString := RandomString(10)
req.Header.Set("X-ROUTING-KEY", randomString)
req.Header.Set("Content-Type", writer.FormDataContentType())

if err != nil {
    fmt.Println(err)
}

// Save response
res, err := client.Do(req)
if err != nil {
    fmt.Println(err)
}
```

5. Apabila Server 2 sudah menerima file dan header dari Server 1, maka pada Server 2 dilakukan:
    - Meng-generate response JSON bahwa proses upload (**bukan proses kompresi**) sudah berhasil/gagal, kemudian mengirimkan response tersebut ke Server 1
    - Meng-execute background Process yang melakukan kompresi file yang telah diterima dari Server 1.
    - Background Process harus membuat koneksi AMQP ke RabbitMQ dengan EXCHANGE_NAME = ***“progress”***
    - Mengirimkan data AMQP menggunakan ROUTING-KEY yang diterima dari Server 1 (pada header X-ROUTING-KEY)

Menghubungkan dengan RabbitMQ

```go
func main() {
    router := gin.Default()

    conn, err := amqp.Dial("amqp://guest:guest@localhost:5672/")
    failOnError(err, "Failed to connect to RabbitMQ")
    defer conn.Close()

    ch, err := conn.Channel()
    failOnError(err, "Failed to open a channel")
    defer ch.Close()

    err = ch.ExchangeDeclare(
        "progress",  // name
        "direct", // type
        true,     // durable
        false,    // auto-deleted
        false,    // internal
        false,    // no-wait
        nil,      // arguments
    )
    failOnError(err, "Failed to declare an exchange")
```

Menerima request dan men-generate JSON bahwa proses upload berhasil

```go
// Handler
router.POST("/compress", func(c *gin.Context) {
    // Retrieve file from request
    file, err := c.FormFile("file")
    if err != nil {
        fmt.Println(c.Request.Body)
        fmt.Println(err)
        c.JSON(http.StatusUnprocessableEntity, gin.H{
            "status": http.StatusUnprocessableEntity,
            "error":  "Unable to parse requeest",
        })
        return
    }

    // Get routing key
    routingKey := c.Request.Header.Get("X-ROUTING-KEY")

    if routingKey == "" {
        fmt.Println("Routing key is empty")
        c.JSON(http.StatusUnprocessableEntity, gin.H{
            "status": http.StatusUnprocessableEntity,
            "error":  "Request header X-ROUTING-KEY is empty",
        })
        return
    }
```

```
c.JSON(http.StatusOK, gin.H{
    "status": "success upload",
})
```

Meng-Execute background proccess

```
// Create new goroutines so it does its task on background
go func() {
    // Open file to be read
    fileContent, err := file.Open()
    if err != nil {
        fmt.Println(err)
        c.JSON(http.StatusUnprocessableEntity, gin.H{
            "status": http.StatusUnprocessableEntity,
            "error":  "Unable to open file",
        })
        return
    }

    // Read bytes of file
    fileBytes, err := ioutil.ReadAll(fileContent)
    if err != nil {
        fmt.Println(err)
        c.AbortWithStatusJSON(http.StatusInternalServerError, gin.H{
            "message": "Unable to convert to bytes",
        })
        return
    }
```

```
// Declare buffer for result of compressed file
var b bytes.Buffer
size := len(fileBytes)
w := gzip.NewWriter(&b)

// Use progress library to track progress change
r := progress.NewWriter(w)

// Get context background
ctx := context.Background()
```

Update progress secara asynchronous dan mengirim ke rabbitmq

n.go

```go
// Start a goroutine again, asynchronously update everytime progress of compress changes
go func() {
    // Declare new ticker
    progressChan := progress.NewTicker(ctx, r, int64(size), 30*time.Millisecond)

    // Wait for ticker update and publish to exchange with routing key from previous
    for _ = range progressChan {
        progresCurrent := (b.Len() * 100 / len(fileBytes) )
        payload := Message{Content: string(strconv.Itoa(progresCurrent)), Status: "In Progress"}
        payloadStr, err := json.Marshal(payload)
        err = ch.Publish(
            "progress", // exchange
            routingKey,      // routing key
            false,   // mandatory
            false,   // immediate
            amqp.Publishing{
                ContentType: "text/plain",
                Body:        []byte(payloadStr),
            })
        failOnError(err, "Failed to publish a message")
    }
    payload := Message{Content: "Saving file", Status: "Saving file"}
    payloadStr, err := json.Marshal(payload)
```

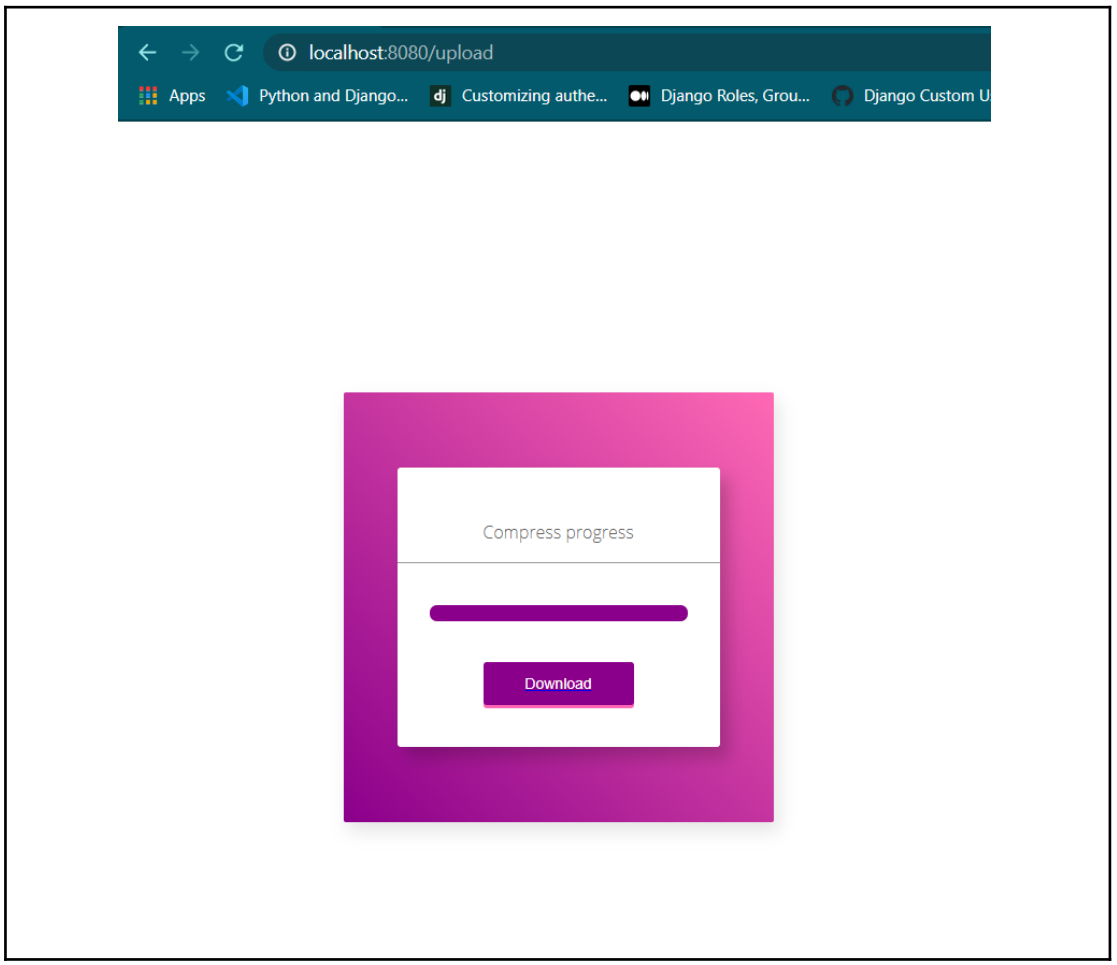
```go
            })
        failOnError(err, "Failed to publish a message")
    }
    payload := Message{Content: "Saving file", Status: "Saving file"}
    payloadStr, err := json.Marshal(payload)
    err = ch.Publish(
        "progress", // exchange
        routingKey,      // routing key
        false,   // mandatory
        false,   // immediate
        amqp.Publishing{
            ContentType: "text/plain",
            Body:        []byte(payloadStr),
        })
    failOnError(err, "Failed to publish a message")
    fmt.Println("\rcompress is completed")
}()
```

Mengirim pesan asinkronus terakhir berupa link file yang dikompres

```go
    // Save to disk
    err = ioutil.WriteFile(file.Filename+".gz", b.Bytes(), 0666)
    if err != nil {
        fmt.Println(err)
        c.AbortWithStatusJSON(http.StatusInternalServerError, gin.H{
            "message": "Unable to compress the file",
        })
        return
    }
    payload := Message{Content: "/compressed" + file.Filename + ".gz", Status: "File saved"}
    payloadStr, err := json.Marshal(payload)
    err = ch.Publish(
        "progress", // exchange
        routingKey,      // routing key
        false,   // mandatory
        false,   // immediate
        amqp.Publishing{
            ContentType: "text/plain",
            Body:        []byte(payloadStr),
        })
    failOnError(err, "Failed to publish a message")
}()
```

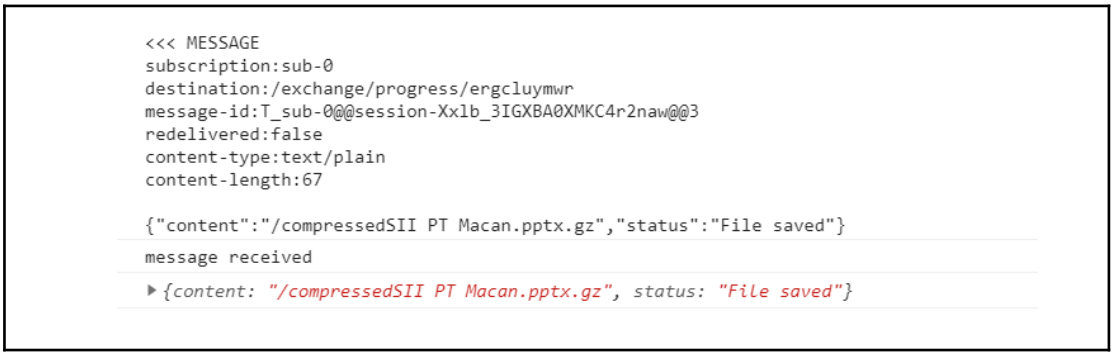
6. Server 1 menghasilkan halaman HTML + JavaScript lengkap dengan koneksi WebSocket & StompJS ke RabbitMQ yang sudah mengarah ke stream ROUTING-KEY yang sudah diaktifkan oleh Background Process. Halaman HTML + Javascript tersebut dikembalikan ke Laptop Mahasiswa.

Tampilan progress



Progress update logging

Saved file logging



Unduh file