L34 exercise-presign

# Share local file with S3 presigned URL

## Exercise

1. Takes a local file and a bucket name as input
2. Uploads the file to the bucket
3. Creates a presigned URL for the file
4. Output the URL to the console
5. Have `main.go` delegate to functions in separate files

## Task 1

- Create `upload.go` file
- Create `s3share.go` file
- Test with `main.go` file

## Task 2

- Create `share.go` file
- Test with `main.go` file

## Task 3

- Update `main.go` file for input parameters

Task - Overview

## Directory structure

```
.
├── go.mod
├── go.sum
├── main
│   └── main.go
├── readme.md
├── s3share.go
├── share.go
├── testdata
│   └── text.txt
└── upload.go
```

- Module name: `s3share`
- Main in its own directory
- `s3share.go` contains init function for the S3 client
- `share.go` contains function to create presigned URL
- `upload.go` contains function to upload file to S3

Task 1 - Step 1

## Init client

- create `module` : `go mod init s3share`
- create `s3share.go`
- add client initialization
- `Client` has a capital `C` to make it public
- handle imports

```go
// file: s3share.go
package s3share

import (
	"context"
	"github.com/aws/aws-sdk-go-v2/config"
	"github.com/aws/aws-sdk-go-v2/service/s3"
)

var Client *s3.Client

func init() {
	// create a s3 client
	cfg, err := config.LoadDefaultConfig(context.TODO())
	if err != nil {
		panic("configuration error, " + err.Error())
	}
	Client = s3.NewFromConfig(cfg)
}
```

Task 1 - Step 2

## Upload function

- create `upload.go`
- create function `Upload`
- use client as parameter for testing (later...)
- return error as last parameter

```go
func Upload(client *s3.Client, filename *string, bucket *string) error {
    return nil
}
```

Task 1 - Step 3

## Upload function

- update function in `upload.go`
- read file into `content`
- populate `PutObjectInput` struct
- (code not shown) handle errors

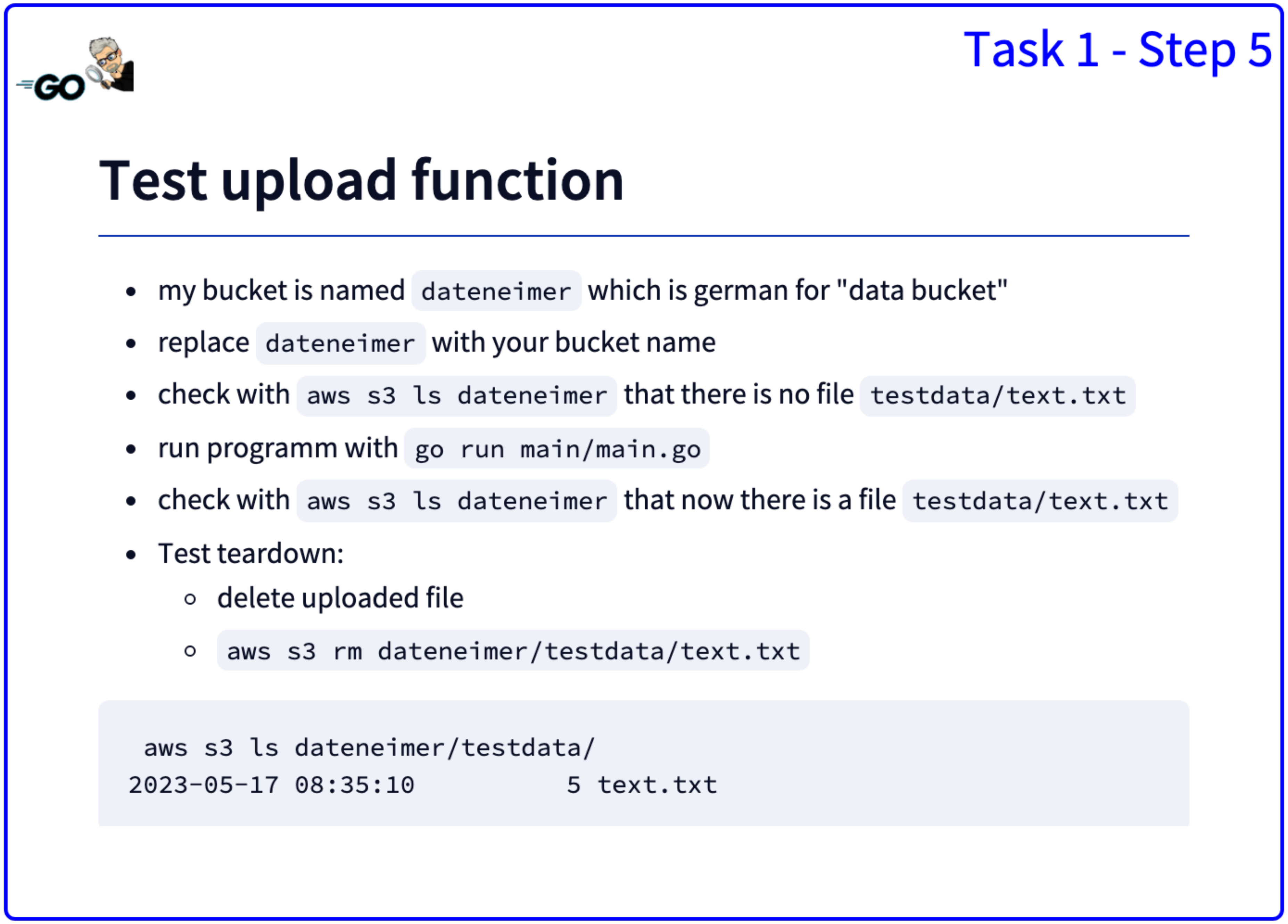
```
	content, err := os.ReadFile(*filename)

	key := filename
	_, err = client.PutObject(
		context.Background(),
		&s3.PutObjectInput{
			Bucket: bucket,
			Key:    key,
			Body:   bytes.NewReader(content),
		},
	)
```

Task 1 - Step 4

## Test upload with main

- update `main/main.go`
- (code not shown) add `package` and `imports`
- create a directory `testdata`
- content of `testdata` is ignored by `go build`
- create a file `testdata/text.txt` with random content

```
func main() {
    // replace "dateneiner" with your bucket name
    bucket := "dateneimer"
    from := "testdata/text.txt"
    err := s3share.Upload(s3share.Client,&from,&bucket)
    if err != nil {
        fmt.Printf("Problem with sharing: %s",err)
        os.Exit(1)
    }
}
```



Task 1 - Step 5

## Test upload function

- my bucket is named `dateneimer` which is german for "data bucket"
- replace `dateneimer` with your bucket name
- check with `aws s3 ls dateneimer` that there is no file `testdata/text.txt`
- run programm with `go run main/main.go`
- check with `aws s3 ls dateneimer` that now there is a file `testdata/text.txt`
- Test teardown:
  - delete uploaded file
  - `aws s3 rm dateneimer/testdata/text.txt`

```
 aws s3 ls dateneimer/testdata/
2023-05-17 08:35:10          5 text.txt
```

Task 2 - Step 6

## Share objects with presigned URLs

- create `share.go`
- create function `Share`
- parameters as with `Upload`

```go
func Share(client *s3.Client, key *string, bucket *string) (string, error) {
    return url, nil
}

```

# Share objects with presigned URLs

- update function `Share` in `share.go`

## Imports

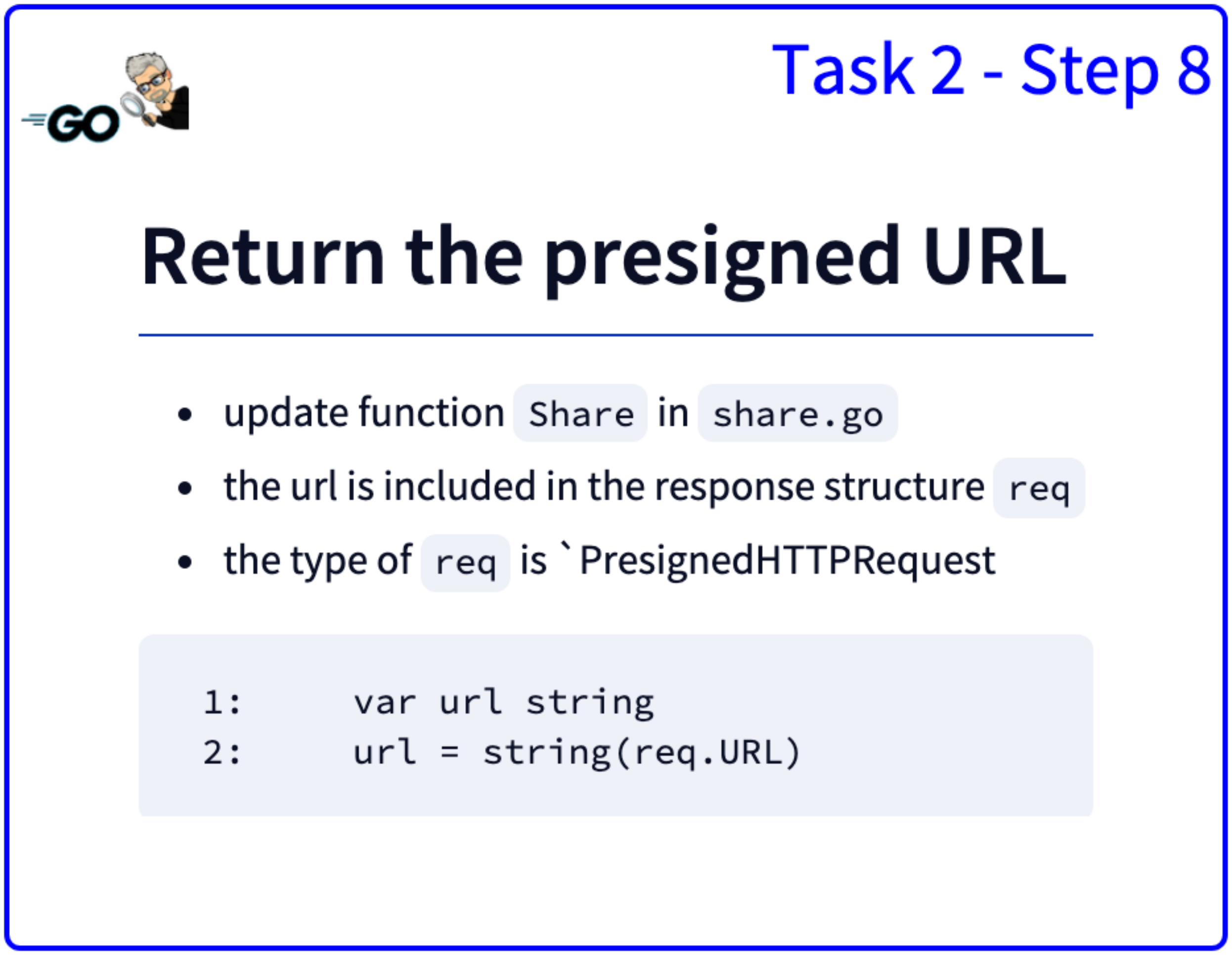
- check that the automated import works
- sometimes it will import v1 from GO SDK, which is wrong
- see line 5

```
1: import (
2:     "context"
3:     "log"
4:     "time"
5:     "github.com/aws/aws-sdk-go-v2/service/s3"
6: )
```

## Create a presigned URL

- you need a `PresignClient` for this
- line 10: set options with variadic function
- the url will be valid `lifetimeSecs` seconds

```
 1:     // Set the expiration time for the presigned URL
 2:     lifetimeSecs := int64(3600)
 3:     s3PresignClient := s3.NewPresignClient(client)
 4:     req, err := s3PresignClient.PresignGetObject(
 5:         context.TODO(),
 6:         &s3.GetObjectInput{
 7:             Bucket: bucket,
 8:             Key:    key,
 9:         },
10:         func(opts *s3.PresignOptions) {
11:             opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
12:         })
13:     if err != nil {
14:         log.Printf("Couldn't get a presigned request to get %v:%v. Here's why: %v\n",
15:             *bucket, *key, err)
16:         return "", err
17:     }
```



Task 2 - Step 8

## Return the presigned URL

- update function `Share` in `share.go`
- the url is included in the response structure `req`
- the type of `req` is `PresignedHTTPRequest

```
    var url string
    url = string(req.URL)
```

Task 2 - Step 9

## Setup test

- update `main.go`
- set fixed parameters
- call `Share` after `Upload`
- print the url

```go
import (
    "fmt"
    "os"
    "s3share"
)

func main() {
    bucket := "dateneimer"
    from := "testdata/text.txt"

    s3share.Upload(s3share.Client,&from, &bucket)
    url, err := s3share.Share(s3share.Client, &from, &bucket)

    if err != nil {
        fmt.Printf("Problem with sharing: %s",err)
        os.Exit(1)
    }
    fmt.Println(url)

}
```

Task 3 - Step 10

## Test share function

- run programm
- you get a presigned url
- copy the url and paste it into a browser
- you should see the content of the file
    - delete uploaded file
    - `aws s3 rm dateneimer/testdata/text.txt`

```
 go run main/main.go
https://dateneimer.s3.eu-central-1.amazonaws.com/testdata/text.txt?X-Amz-Algorithm=A...ure=88...65218
```

Congratulations! You have finished the exercise.

Task 3 - Step 11 optional

## Update main go to use flags

If in doubt, check the source in:

```
go-on-aws-source/aws-go-sdk-v2/L34-exercise-presign/code-task3
```