L29-exercise-list-ec2-instances

List EC2 instances

Task 1: main.go

1. List the first 10 EC2 instances with ID and status
2. Gives also the total number of instances
3. Uses paging to minimize the ammount of traffic for the requests

Task 2: refactor

4. refactor client creation in init function
5. move the listing in its own package

Task 1 - Step 1

Create configuration

- create `main.go`
- add into main function

```
1:      cfg, err := config.LoadDefaultConfig(context.TODO())
2:      if err != nil {
3:          panic("configuration error, " + err.Error())
4:      }
5:      client := ec2.NewFromConfig(cfg)
```

Task 1 - Step 2

Create Input parameter

- add after configuration
- let editor handle imports

```
1:      parms := &ec2.DescribeInstancesInput{
2:          MaxResults: aws.Int32(10),
3:      }
```

Task 1 - Step 3

init paginator

- init an `DescribeInstancesPaginator`
- set page counter to initial value

```
1:      paginator := ec2.NewDescribeInstancesPaginator(client, parms)
2:      pagecounter := 1
```

Task 1 - Step 4

Paginate and print results

- 1: `for` loop with paginator (kind of while-do)
- 2: get next page
- 8-15: range instances

```
 1:     for paginator.HasMorePages() {
 2:         page, err := paginator.NextPage(context.TODO())
 3:         if err != nil {
 4:             fmt.Print("Error calling ec2: ", err)
 5:         }
 6:         fmt.Printf("Page: %v\n", pagecounter)
 7:         pagecounter += 1
 8:         for _, reservation := range page.Reservations {
 9:             for k, instance := range reservation.Instances {
10:                 fmt.Printf("Instance number: %v, ID: %v, Status: %v \n",
11:                     k, *instance.InstanceId, instance.State.Name,
12:                 )
13:             }
14:         }
15:     }
```

Task 1 - Step 5

Run and test

1. launch an small ec2 instance, remember to terminate afterwards
2. `go run .`
3. Output should be like (with other Status):

```
Page: 1
Instance number: 0, ID: i-07438a1ac027acdb9, Status: terminated
```

Main.go

```
 1: package main
 2: import (
 3:     "context"
 4:     "fmt"
 5:
 6:     "github.com/aws/aws-sdk-go-v2/aws"
 7:     "github.com/aws/aws-sdk-go-v2/config"
 8:     "github.com/aws/aws-sdk-go-v2/service/ec2"
 9: )
10:     cfg, err := config.LoadDefaultConfig(context.TODO())
11:     if err != nil {
12:         panic("configuration error, " + err.Error())
13:     }
14:     client := ec2.NewFromConfig(cfg)
15:     parms := &ec2.DescribeInstancesInput{
16:         MaxResults: aws.Int32(10),
17:     }
18:     paginator := ec2.NewDescribeInstancesPaginator(client, parms)
19:     pagecounter := 1
20:     for paginator.HasMorePages() {
21:         page, err := paginator.NextPage(context.TODO())
22:         if err != nil {
23:             fmt.Print("Error calling ec2: ", err)
24:         }
25:         fmt.Printf("Page: %v\n", pagecounter)
26:         pagecounter += 1
27:         for _, reservation := range page.Reservations {
28:             for k, instance := range reservation.Instances {
29:                 fmt.Printf("Instance number: %v, ID: %v, Status: %v \n",
30:                     k, *instance.InstanceId, instance.State.Name,
31:                 )
32:             }
33:         }
34:     }
```

Task 2 - Step 6: lister.go

Create file

`lister.go`

Task 2 - Step 7

Add package

```
1: package instancelister
```

Task 2 - Step 8

Manage imports

- context for api calls
- fmt for printing
- aws for pointers
- config for client configuration
- service/ec2 for ec2 api

```
1: import (
2:     "context"
3:     "fmt"
4:
5:     "github.com/aws/aws-sdk-go-v2/aws"
6:     "github.com/aws/aws-sdk-go-v2/config"
7:     "github.com/aws/aws-sdk-go-v2/service/ec2"
8: )
```

Task 2 - Step 9

Create client

- Create a global variable `Client`
- Capital "C"
- We can use this für unit testing later to inject mock clients
- move client initialzation into init(), so its called automatically

```
 1: var Client *ec2.Client
 2:
 3: func init() {
 4:     cfg, err := config.LoadDefaultConfig(context.TODO())
 5:     if err != nil {
 6:         panic("configuration error, " + err.Error())
 7:     }
 8:     Client = ec2.NewFromConfig(cfg)
 9:
10: }
```

Task 2 - Step 10

Create struct for results

- You don not want whole results
- Bundle relevant data in struct

```
1: type Instances struct {
2:     Name string
3:     Status string
4: }
```

Task 2 - Step 11

Refactor lister in exported function

- Return slice of your custom struct
- Return error as second type
- initialize your return variable

```
1: func ListInstances(client *ec2.Client) ([]*Instances, error) {
2:     instances := []*Instances{}
```

Task 2 - Step 12

Move paginator loop into `ListInstances`

- 12: Return error if a problem occurs
- 22: append each result to your return variable
- 26: return result and nil error

```
 1:     parms := &ec2.DescribeInstancesInput{
 2:         MaxResults: aws.Int32(10),
 3:     }
 4:
 5:     paginator :=  ec2.NewDescribeInstancesPaginator(client, parms)
 6:     pagecounter := 1
 7:
 8:     for paginator.HasMorePages() {
 9:         page, err := paginator.NextPage(context.TODO())
10:         if err != nil {
11:             fmt.Print("Error calling ec2: ", err)
12:             return nil, err
13:         }
14:         fmt.Printf("Page: %v\n",pagecounter)
15:         pagecounter+=1
16:         for _, reservation := range page.Reservations {
17:             for _, instance := range reservation.Instances {
18:                 newinstance := &Instances{
19:                     Name: *instance.Tags[0].Value,
20:                     Status: string(instance.State.Name),
21:                 }
22:                 instances = append(instances, newinstance)
23:             }
24:         }
25:     }
26:     return instances, nil
27: }
```

Task 2 - Step 13

Create new main

- create `main.go` in subdirectory `main`
- 4: You import your "importlister"
- 10: use the `Client` from `instancelister`, the `init` function will automatically create it

```
 1: package main
 2:
 3: import (
 4:     "instancelister"
 5:     "fmt"
 6:
 7: )
 8:
 9: func main() {
10:     instances, err := instancelister.ListInstances(instancelister.Client)
11:     if err != nil {
12:         fmt.Print("Error calling instancelister: ", err)
13:     }
14:     for i,instance := range instances {
15:         fmt.Printf("Instance number: %v, ID: %v, Status: %v \n",
16:             i+1,instance.Name,instance.Status,
17:         )
18:
19:     }
20: }
```

Task 2 - Step 14

Test refactored program

- run with `go run main/main.go`
- You should see the same result as in task1.