# Uno dates difference calculator challenge.

1. **Problem statement sent by uno:**

**CODING CHALLENGE AT UNO**

Deploy a microservice using ECS or EKS (AWS).

Microservice should have a single HTTP endpoint. Request should provide two date values:

- fromDate (DD.MM.YYYY)
- toDate (DD.MM.YYYY)

Response should be the number of days between the two dates (exclusive). Both dates will be greater than 01.01.1900. Dates may be in the future.

- The result for each call should be stored in a DynamoDB table.
- Code stored in a public BitBucket or GitHub repo.
- Use Terraform or CloudFormation for infrastructure.

Provide instructions for how to deploy into a new AWS account. Also provide instructions on how to test the endpoint. Do not worry about identity and/or access.

- Code should include unit tests.

*Please explicitly calculate the number of days between the two dates - do not use any built-in functions.*

Example 1

fromDate = 01.01.2020
toDate = 03.01.2020
daysDiff = 1

Example 2

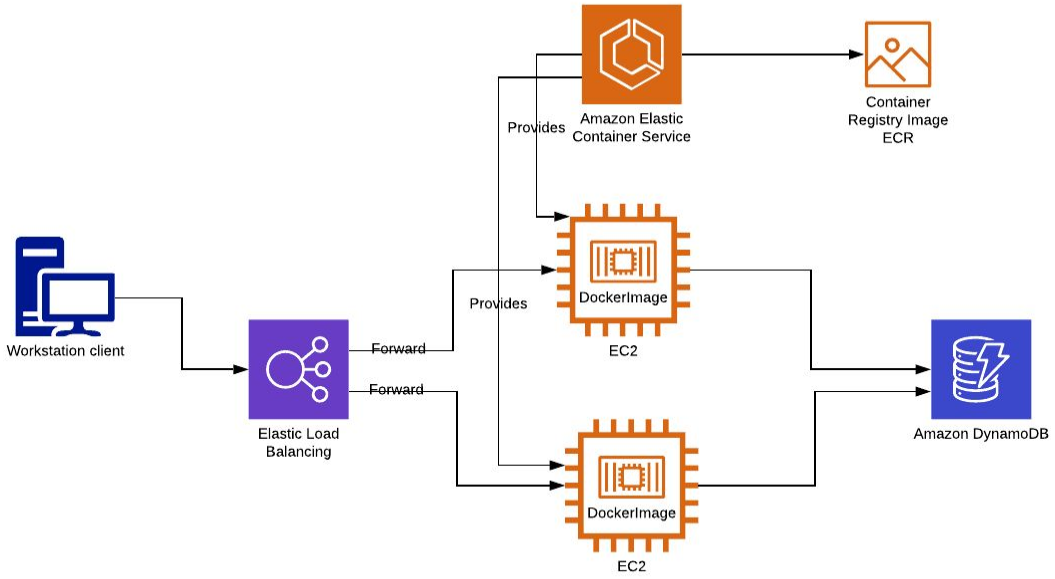
fromDate = 01.01.2020
toDate = 01.01.2030
daysDiff = 3652

## 2. Solution

In order to solve the above coding challenge, I have used the following technologies:

- Java 8
- SpringBoot
- Terraform
- DynamoDB
- JUnit ( unit + integration tests )
- Jacoco ( code coverage )

The application architecture consists on the following components as shown the diagram below:



## 3. Testing the application

In order to test the application all you need is to send a simple GET request to the load balancer, as follow:

```
curl
"http://uno-ecs-load-balancer-1411810678.us-east-1.elb.amazonaws.com/calcu
late/dates-diff?startDate=01.01.2020&endDate=01.01.2030"
```

Result: 3652

You can use either the command line, or a more sophisticated tool, such as Postman : https://www.postman.com/

## 4. Reproducing the environment

First of all, we need to setup our local environment to be able to deploy both infra and the application. These are the steps to setup your environment and deploy to a brand new AWS account.

1. Setup your aws account : https://aws.amazon.com/premiumsupport/knowledge-center/create-and-activate-aws-account/
2. Install the AWS CLI ( command line tools ) : https://aws.amazon.com/cli/
3. Configure the AWS CLI : https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html
4. Install terraform: https://www.terraform.io/downloads.html
5. Install Docker : https://www.docker.com/get-started
6. Install GIT / configure git : https://git-scm.com/
7. Clone the app → git clone https://marciomarinho@bitbucket.org/marciomarinho/uno-dates-diff-calculator.git
8. Clone the infra → git clone https://marciomarinho@bitbucket.org/marciomarinho/uno-ecs-cluster.git

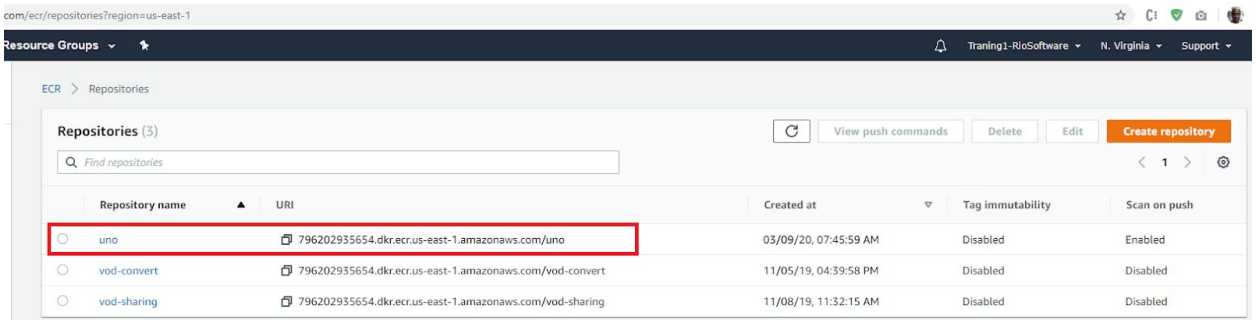
The next step once you get all the above installed and configured, is to create an ECR using the uno-ecs-cluster terraform project.

Open a command line and to the terraform project root folder, e.g:
cd d:\projects\interview\uno\uno-ecs-cluster

and run:

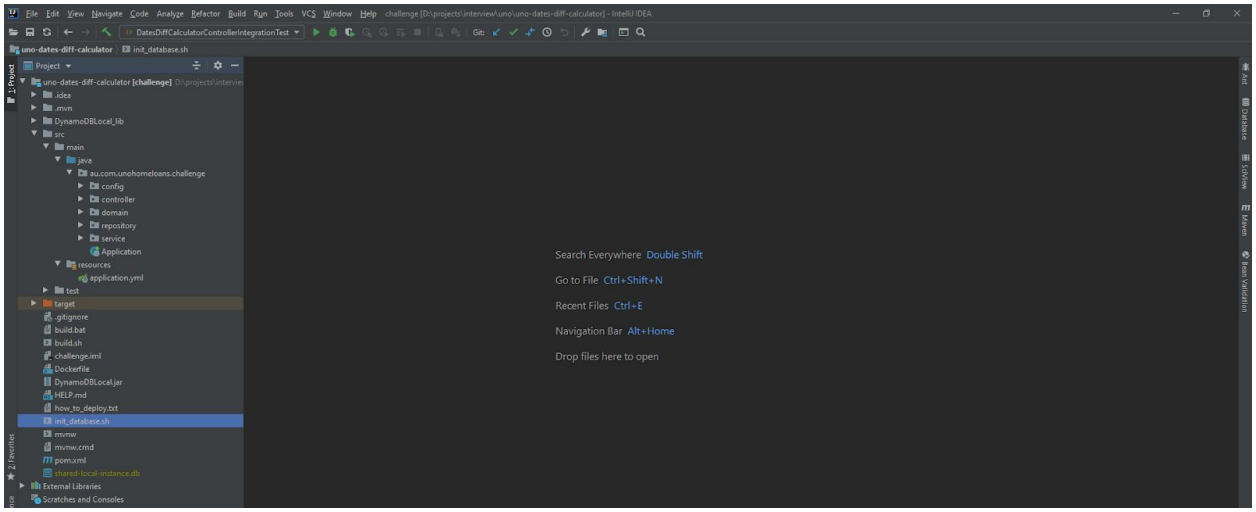
```
terraform init
terraform apply -target=module.ecr
Ps : When prompted type yes to confirm stack creation.
```

Which will initialize the terraform project and create the Docker container repository to host our docker images.



Now, it is time to run locally and/or build our application.

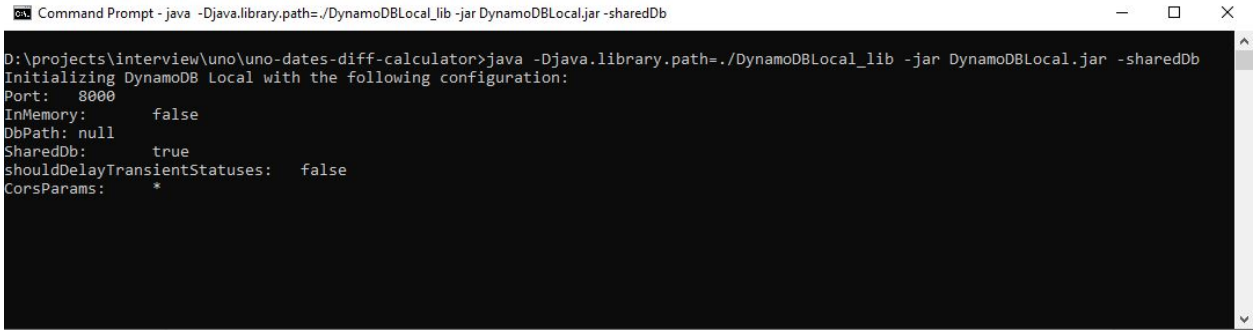
The application has been built using maven, so the structure is pretty simple and should sound familiar. You can also import in on your preferred IDE:




In order to run it locally you will need to first start dynamodb locally, which is included in the project. Just open a new terminal/command window, go to the project root folder, type and press enter:

**java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -sharedDb**

You should see DynamoDB running locally:



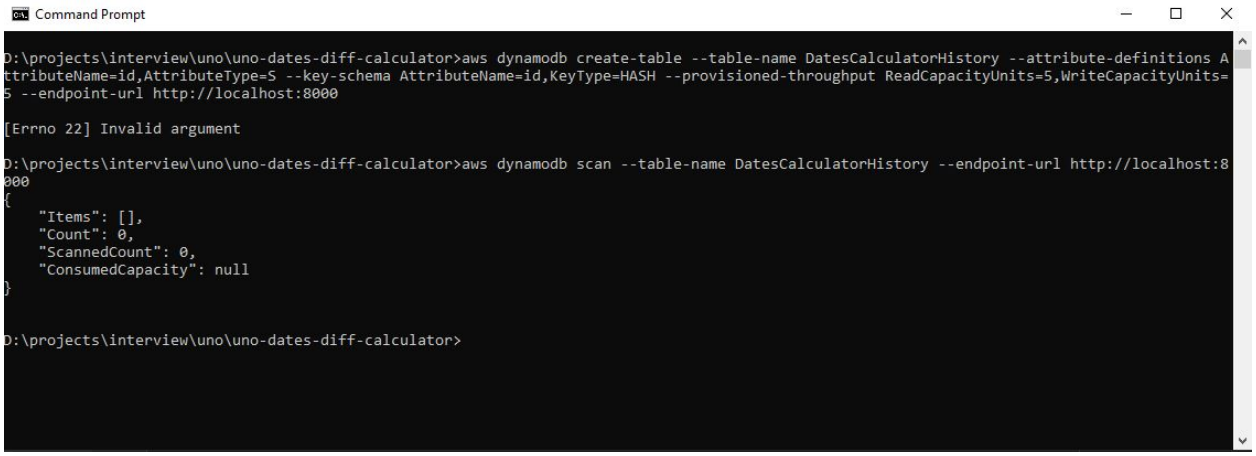

You should create the schema next:

```
aws dynamodb create-table --table-name DatesCalculatorHistory
--attribute-definitions AttributeName=id,AttributeType=S --key-schema
AttributeName=id,KeyType=HASH --provisioned-throughput
ReadCapacityUnits=5,WriteCapacityUnits=5 --endpoint-url http://localhost:8000
```

You can also drop / list items using the CLI ( if needed )

```
aws dynamodb scan --table-name DatesCalculatorHistory --endpoint-url
http://localhost:8000
aws dynamodb delete-table --table-name DatesCalculatorHistory --endpoint-url
http://localhost:8000
```

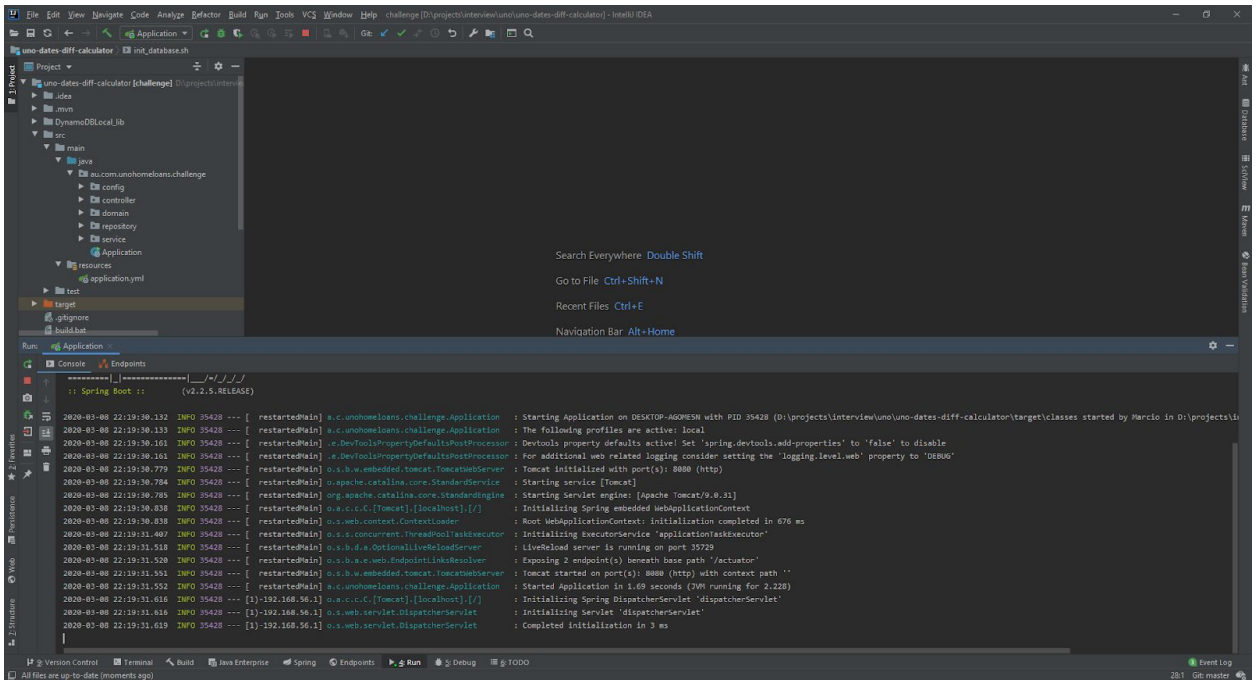
Just open another command window and past the command there.




Sometimes the CLI throws some random errors, but you can ignore them as you can see in the above picture. The local table has been created.

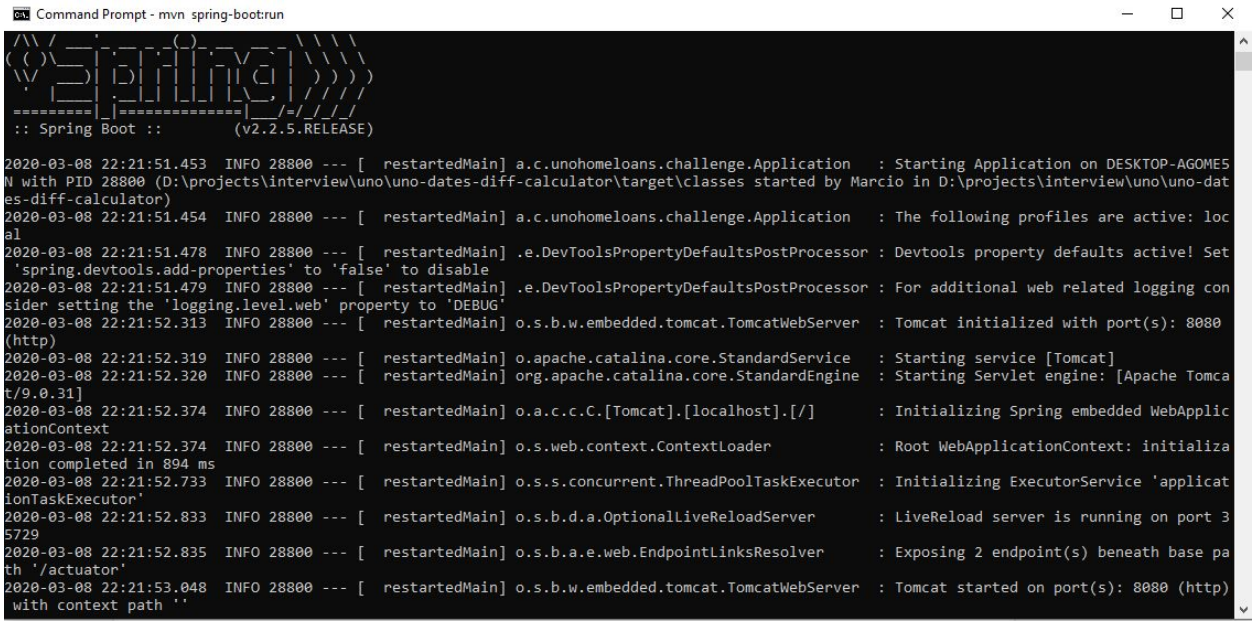
## Running the application.

Now, all you need is to run the application either from your IDE or using the command line.



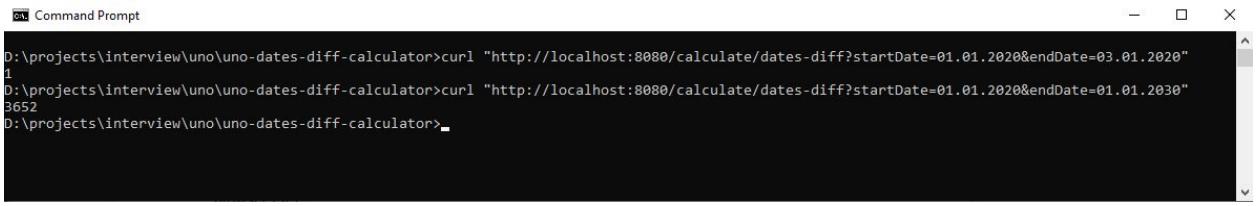

Alternatively, you can run the app using a new command line window and executing the following command:

mvn spring-boot:run



Once the application is up and running, testing is as simple as opening another window and typing a curl command. You can also use any other tool that is able to send a http request ( postman, browser, etc).

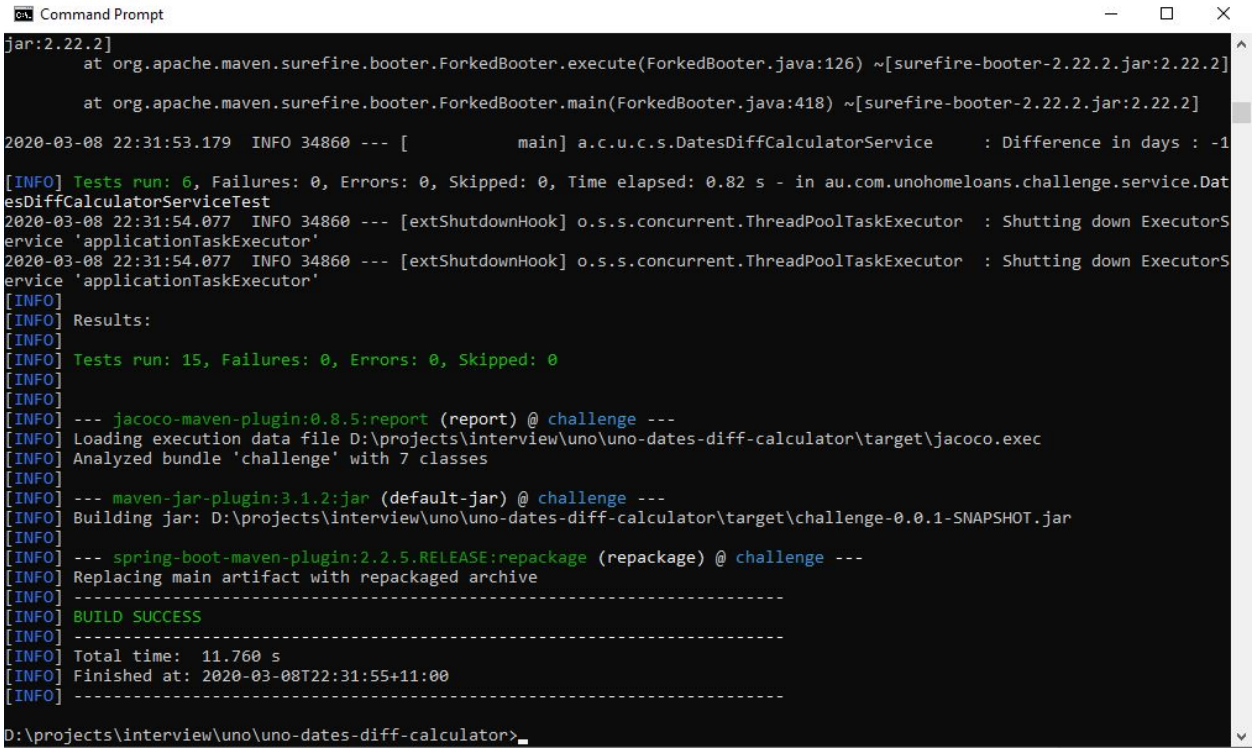
```
D:\projects\interview\uno\uno-dates-diff-calculator>curl
"http://localhost:8080/calculate/dates-diff?startDate=01.01.2020&endDate=03.01.2020"
1

D:\projects\interview\uno\uno-dates-diff-calculator>curl
"http://localhost:8080/calculate/dates-diff?startDate=01.01.2020&endDate=01.01.2030"
3652
```




Now that our application is running locally we can proceed on building it, adding a docker container and then push to our ECR.

Stop the application, and in the command line run :

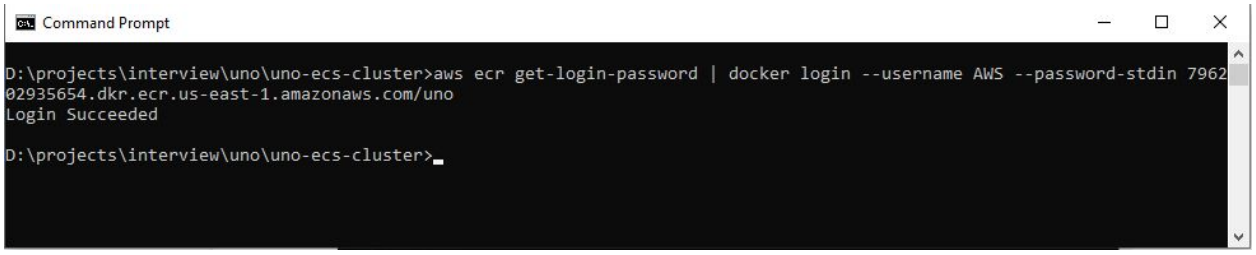
```
mvn clean package
```



```
Command Prompt
jar:2.22.2]
        at org.apache.maven.surefire.booter.ForkedBooter.execute(ForkedBooter.java:126) ~[surefire-booter-2.22.2.jar:2.22.2]

        at org.apache.maven.surefire.booter.ForkedBooter.main(ForkedBooter.java:418) ~[surefire-booter-2.22.2.jar:2.22.2]

2020-03-08 22:31:53.179  INFO 34860 --- [           main] a.c.u.c.s.DatesDiffCalculatorService     : Difference in days : -1

[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.82 s - in au.com.unohomeloans.challenge.service.Dat
esDiffCalculatorServiceTest
2020-03-08 22:31:54.077  INFO 34860 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor  : Shutting down ExecutorS
ervice 'applicationTaskExecutor'
2020-03-08 22:31:54.077  INFO 34860 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor  : Shutting down ExecutorS
ervice 'applicationTaskExecutor'
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 15, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.5:report (report) @ challenge ---
[INFO] Loading execution data file D:\projects\interview\uno\uno-dates-diff-calculator\target\jacoco.exec
[INFO] Analyzed bundle 'challenge' with 7 classes
[INFO]
[INFO] --- maven-jar-plugin:3.1.2:jar (default-jar) @ challenge ---
[INFO] Building jar: D:\projects\interview\uno\uno-dates-diff-calculator\target\challenge-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.2.5.RELEASE:repackage (repackage) @ challenge ---
[INFO] Replacing main artifact with repackaged archive
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  11.760 s
[INFO] Finished at: 2020-03-08T22:31:55+11:00
[INFO] ------------------------------------------------------------------------

D:\projects\interview\uno\uno-dates-diff-calculator>
```

**Login on ECR :**
**More info → https://aws.amazon.com/ecr/**

**aws ecr get-login-password | docker login --username AWS --password-stdin 796202935654.dkr.ecr.us-east-1.amazonaws.com/uno**
**Replace the above account number by yours.**



```
Command Prompt
D:\projects\interview\uno\uno-ecs-cluster>aws ecr get-login-password | docker login --username AWS --password-stdin 7962
02935654.dkr.ecr.us-east-1.amazonaws.com/uno
Login Succeeded

D:\projects\interview\uno\uno-ecs-cluster>
```

**docker build uno .**

```
Command Prompt

D:\projects\interview\uno\uno-dates-diff-calculator>docker build -t uno .
Sending build context to Docker daemon  88.28MB
Step 1/4 : FROM openjdk:8-jdk-alpine
 ---> a3562aa0b991
Step 2/4 : ARG JAR_FILE=target/*.jar
 ---> Using cache
 ---> 24f6663c29bc
Step 3/4 : COPY ${JAR_FILE} dates-calculator.jar
 ---> b9bb9732ece7
Step 4/4 : ENTRYPOINT ["java","-jar","/dates-calculator.jar", "-Dspring.profiles.active=${SPRING_PROFILES_ACTIVE}"]
 ---> Running in 8f9b943df817
Removing intermediate container 8f9b943df817
 ---> 0d878c0419e4
Successfully built 0d878c0419e4
Successfully tagged uno:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories
added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensi
tive files and directories.

D:\projects\interview\uno\uno-dates-diff-calculator>
```

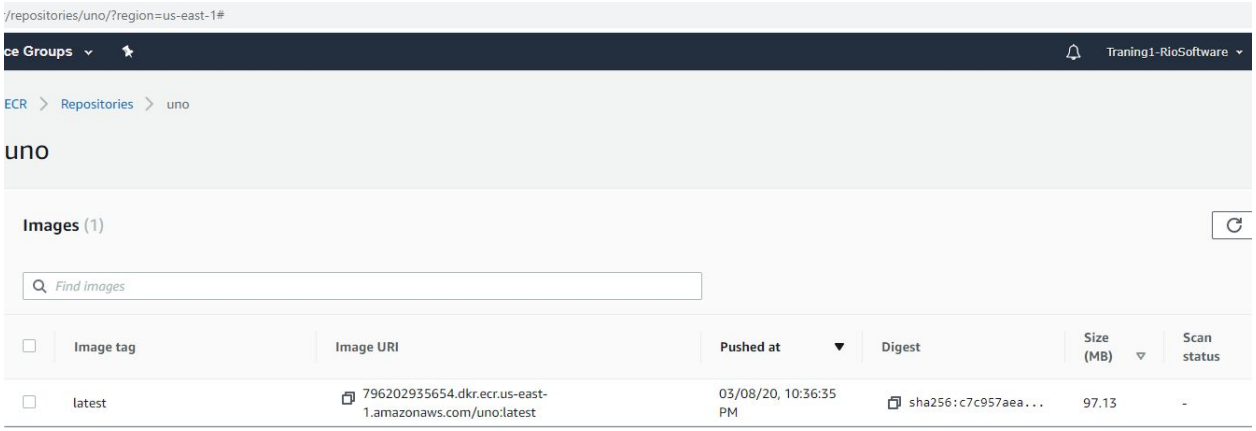
docker tag uno:latest XXXXXXXXXXXX.dkr.ecr.us-east-1.amazonaws.com/uno:latest

docker push XXXXXXXXXXXX.dkr.ecr.us-east-1.amazonaws.com/uno:latest

- Note that you have to use your account number here.

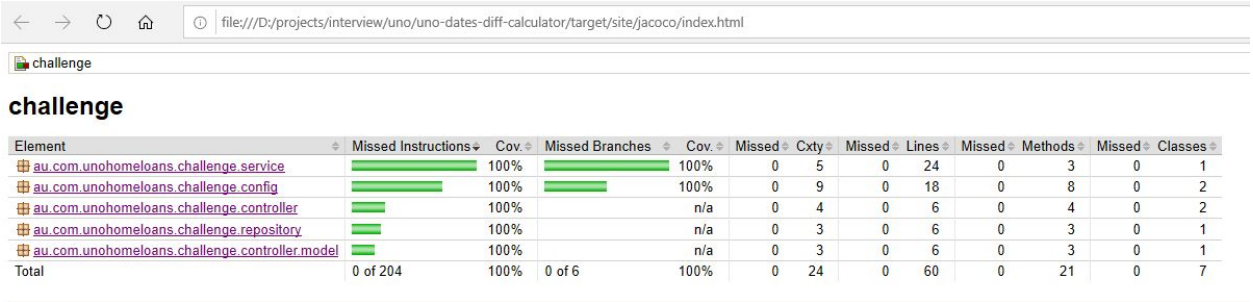
```
Command Prompt

D:\projects\interview\uno\uno-dates-diff-calculator>docker tag uno:latest 796202935654.dkr.ecr.us-east-1.amazonaws.com/uno:l
atest

D:\projects\interview\uno\uno-dates-diff-calculator>docker push 796202935654.dkr.ecr.us-east-1.amazonaws.com/uno:latest
The push refers to repository [796202935654.dkr.ecr.us-east-1.amazonaws.com/uno]
ba762f4df702: Pushed
ceaf9e1ebef5: Layer already exists
9b9b7f3d56a0: Layer already exists
f1b5933fe4b5: Layer already exists
latest: digest: sha256:c7c957aea30968f538813862b0bf8279e17b98a29ad70a18a8f44fc0d2c52521 size: 1159

D:\projects\interview\uno\uno-dates-diff-calculator>
```

If you go to ECR console, you should see your brand new image there:



## Tests Coverage

The tests coverage can be found inside the folder :
D:\projects\interview\uno\uno-dates-diff-calculator\target\site\jacoco
file:///D:/projects/interview/uno/uno-dates-diff-calculator/target/site/jacoco/index.html




| Element | Missed Instructions | Cov. | Missed Branches | Cov. | Missed | Cxty | Missed | Lines | Missed | Methods | Missed | Classes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| au.com.unohomeloans.challenge.service | | 100% | | 100% | 0 | 5 | 0 | 24 | 0 | 3 | 0 | 1 |
| au.com.unohomeloans.challenge.config | | 100% | | 100% | 0 | 9 | 0 | 18 | 0 | 8 | 0 | 2 |
| au.com.unohomeloans.challenge.controller | | 100% | | n/a | 0 | 4 | 0 | 6 | 0 | 4 | 0 | 2 |
| au.com.unohomeloans.challenge.repository | | 100% | | n/a | 0 | 3 | 0 | 6 | 0 | 3 | 0 | 1 |
| au.com.unohomeloans.challenge.controller.model | | 100% | | n/a | 0 | 3 | 0 | 6 | 0 | 3 | 0 | 1 |
| Total | 0 of 204 | 100% | 0 of 6 | 100% | 0 | 24 | 0 | 60 | 0 | 21 | 0 | 7 |

## Deploying the infrastructure and application to AWS.

**Note : I'm using the aws region us-east-1.**

Open the terraform project using your preferred text editor/ide and change the docker image repository address inside the file task-definition-app.json, as it must contain **YOUR** repository address. Btw, this piece could also be automated, but I decided to leave it this way for simplicity.

```
[
 {
   "name": "dates-calculator",
   "image": "796202935654.dkr.ecr.us-east-1.amazonaws.com/uno:latest",
   "cpu": 10,
   "memory": 512,
   "essential": true,
   "portMappings": [
     {
       "containerPort": 8080,
       "hostPort": 8080
     }
   ],
   "logConfiguration": {
     "logDriver": "awslogs",
     "options": {
       "awslogs-group": "awslogs-calculator",
       "awslogs-region": "us-east-1",
       "awslogs-stream-prefix": "awslogs-uno-calculator"
     }
   },
   "environment" : [
     { "name" : "SPRING_PROFILES_ACTIVE", "value" : "prod"}
   ]
 }
]
```

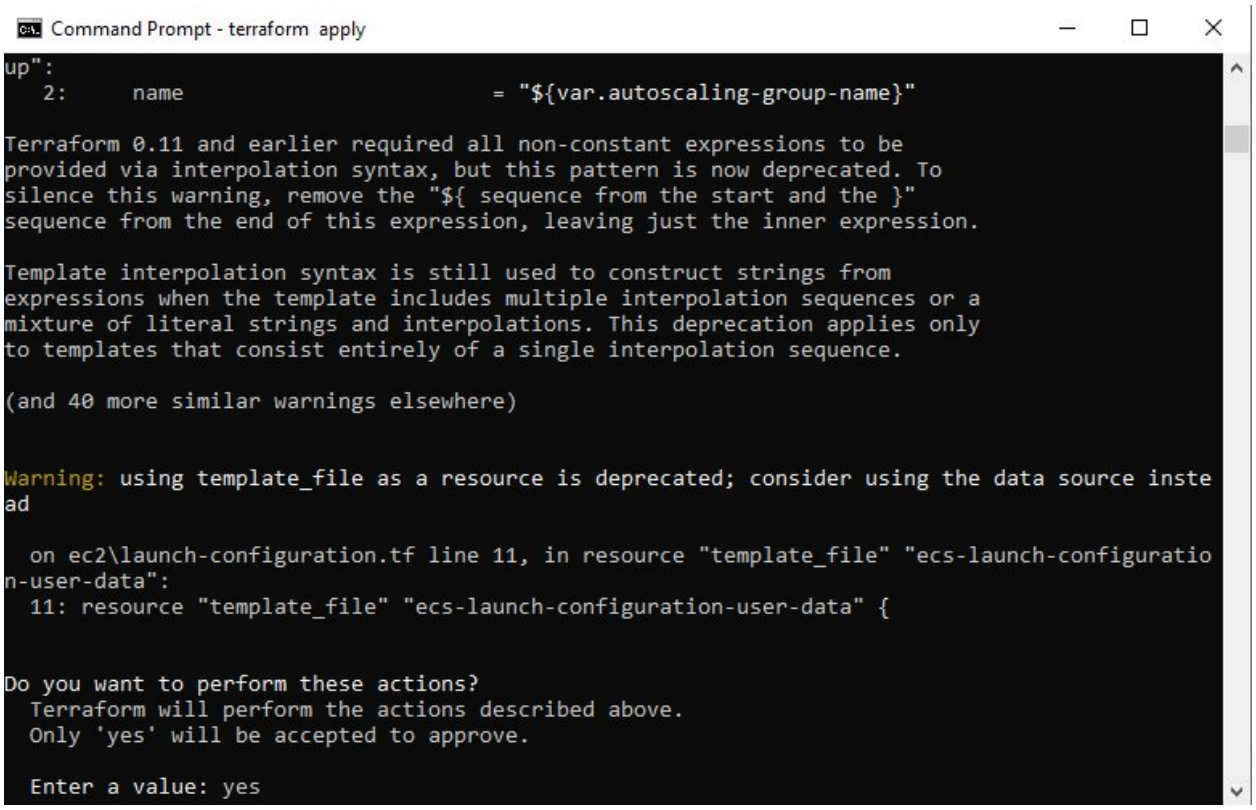
After that go back to the command line on the terraform project root folder, e.g:

**cd d:\projects\interview\uno\uno-ecs-cluster**

Run the commands in the **following order** to satisfy all creation dependencies :

terraform apply -target=module.ecr
terraform apply -target=module.vpc
terraform apply -target=module.iam
terraform apply -target=module.database
terraform apply -target=module.ec2
terraform apply -target=module.ecs

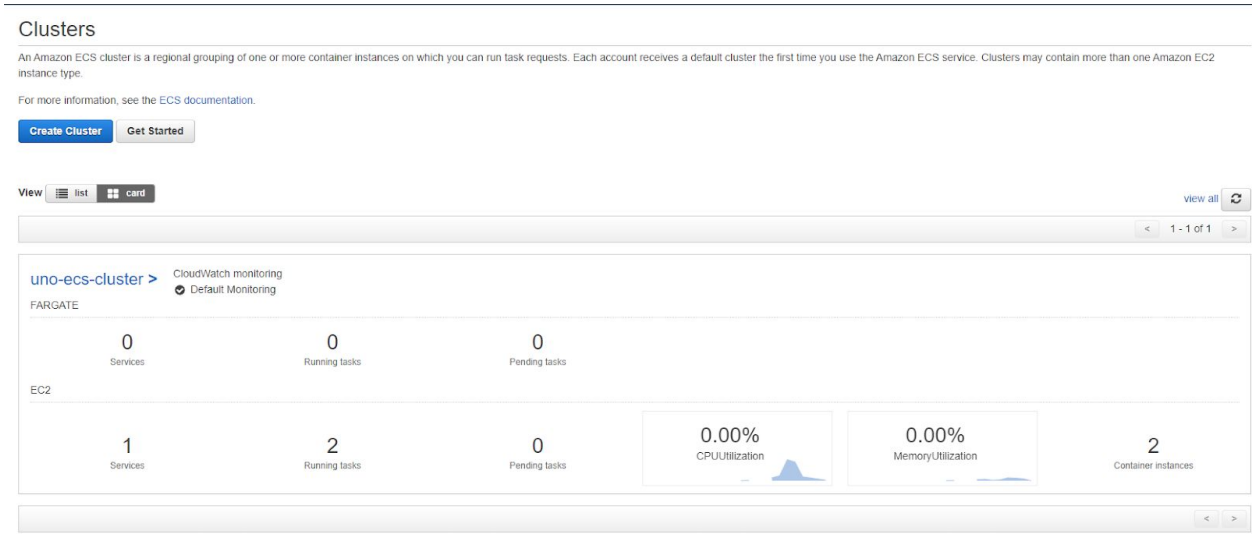
( again, type yes to confirm creation on each step )



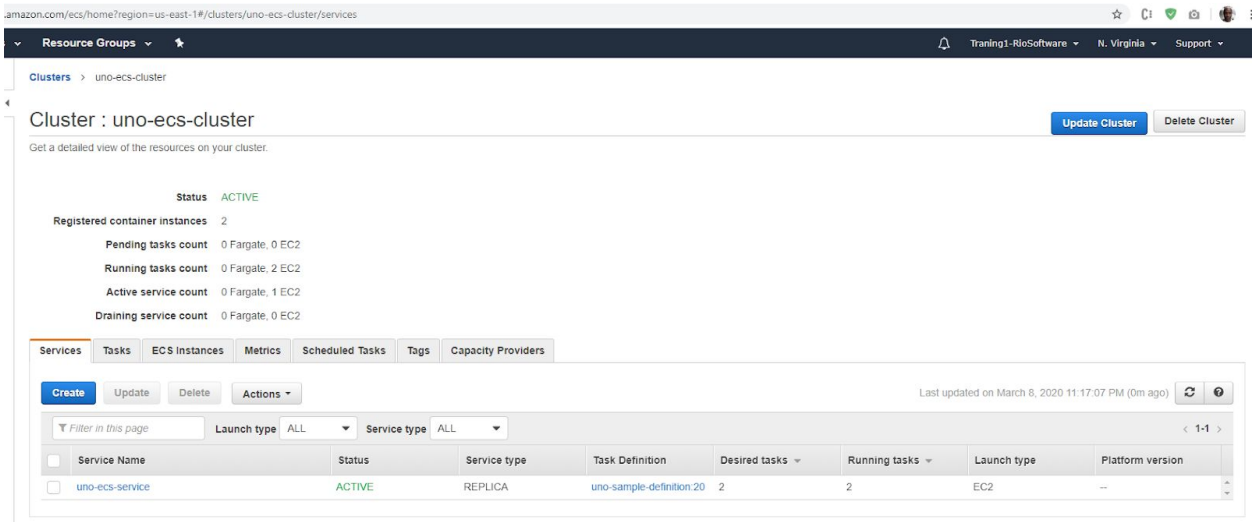
## AWS Environment

Just go to the ECS console and double check you have :

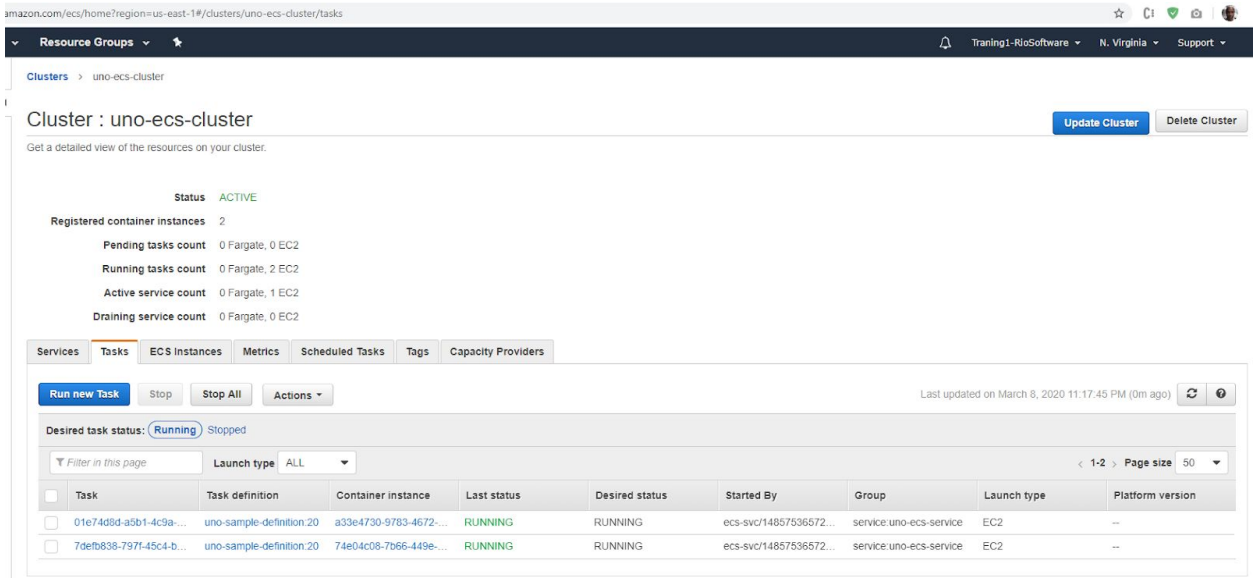
1) Your cluster with 2 instances ( or any other parameter you may have changed )



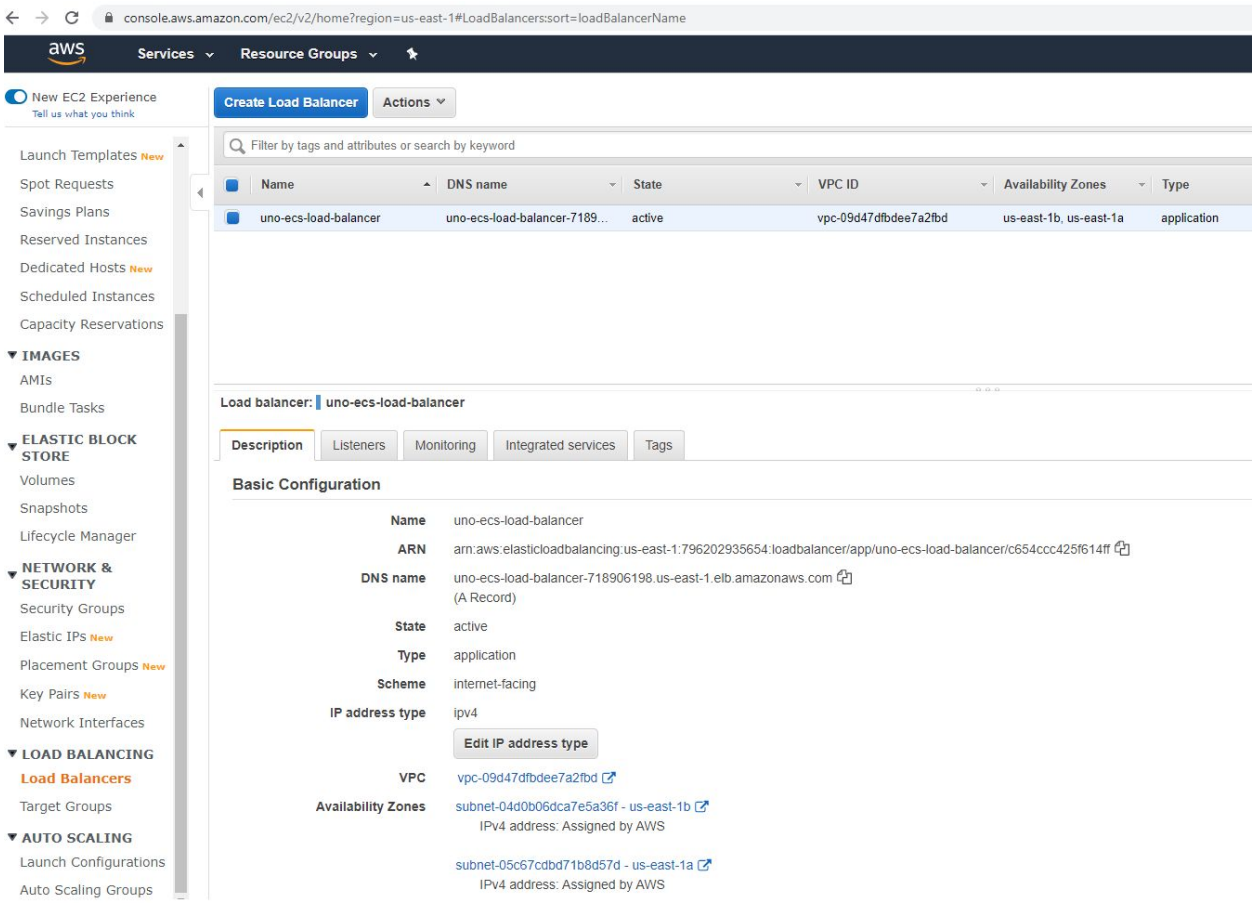

2) You should have one service inside your cluster:



3) You should have 2 tasks running:




4) Go to the EC2 console and check your load balancer:



5) Go to DynamoDB console and check your table:


## We are now ready to test our application in production:

Just copy your load balancer DNS name and add the application path.

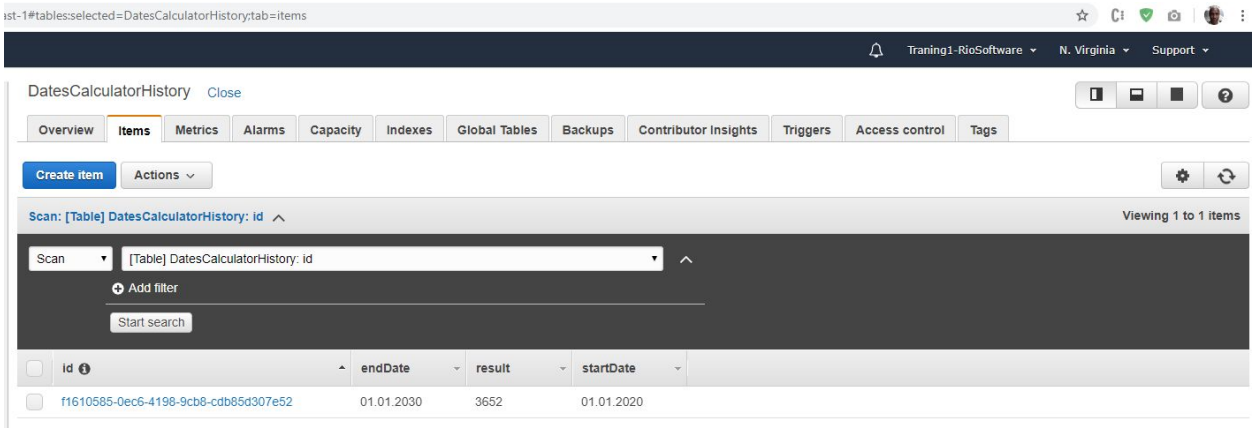
This is my production load balancer :

**uno-ecs-load-balancer-1411810678.us-east-1.elb.amazonaws.com**

curl
“http://uno-ecs-load-balancer-1411810678.us-east-1.elb.amazonaws.com/dates-diff?startDate=01.01.2020&endDate=01.01.2030”

response : 3652

You should also see a new entry on DynamoDB for each request.




Congratulations, you did it!

Please, don’t hesitate in contacting me if you have any trouble setting up your environment.