Amazon Web Services Tutorials

January 17, 2019

Abstract

This document aims to help groups working on the Feniks project to understand what are Amazon Web Services and how to use them.

Each services will be introduced briefly and the different steps to create and manage them will be explained with screenshots.

Contents

1 What is Amazon Web Services?

1.1 Introduction

Amazon Web Services(AWS) is a cloud service from Amazon, which provides services in the form of building blocks, these building blocks can be used to create and deploy any type of application in the cloud.

1.2 Which services will be used for the project?

We will use the following services to create, maintain and deploy the Feniks application:

- **S3 bucket**: storage service
- **API Gateway**: manage restful API
- **Lambda Functions**: enable to create a serverless application
- **DynamoDB**: non-relational database

2 AWS CLI

2.1 Introduction

The AWS Command Line Interface (CLI) is a unified tool to manage your AWS services from a terminal. It enables you to control and configure multiple AWS services from the command line and automate them through scripts.

2.2 How to install AWS CLI?

Command to install awscli on Mac with Homebrew.

```
$ brew install awscli
```

2.3 How to configure aws CLI?

After installing AWS CLI, you need to Add your Access Key ID and Secret Access Key. In order to so, enter the following commands:

```
$ aws configure
```

You will be asked to provide the AWS Access Key ID...

```
AWS Access Key ID [None]:
```

... and the AWS Secret Access Key

```
AWS Secret Access Key:
```

You can leave the Default region name and Default output format blank by clicking on Enter

2.4 Structure of AWS CLI

The basic command line structure:

```
aws <command> <subcommand> [options and parameters*]
```

Example of a command.
This command will list all the S3 buckets:

```
aws s3 ls
```

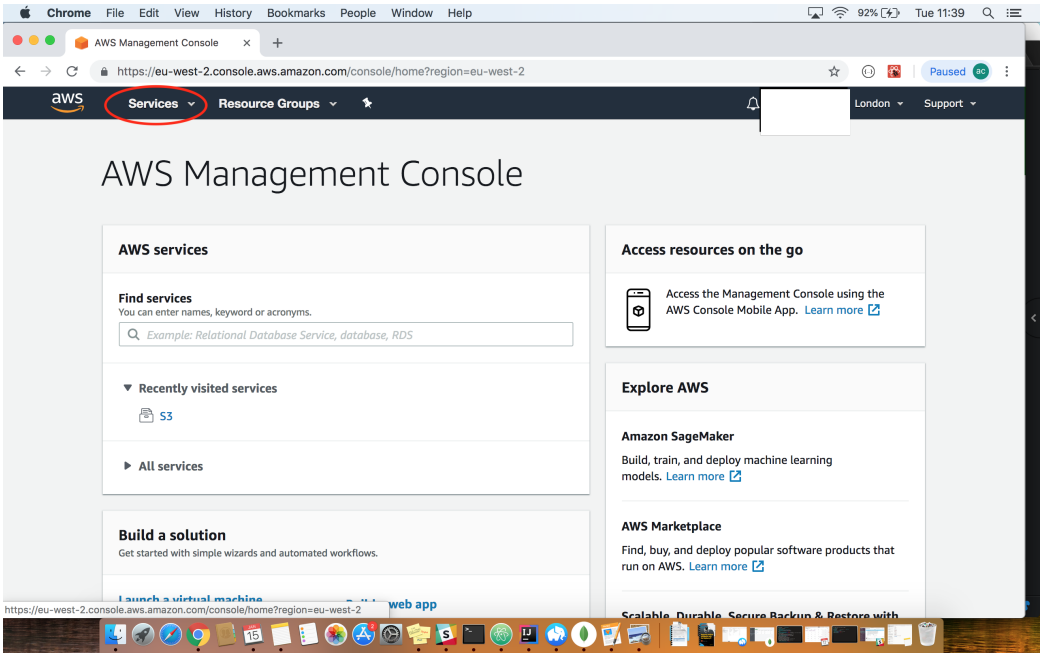
2.5 Ressources

3 S3 Bucket

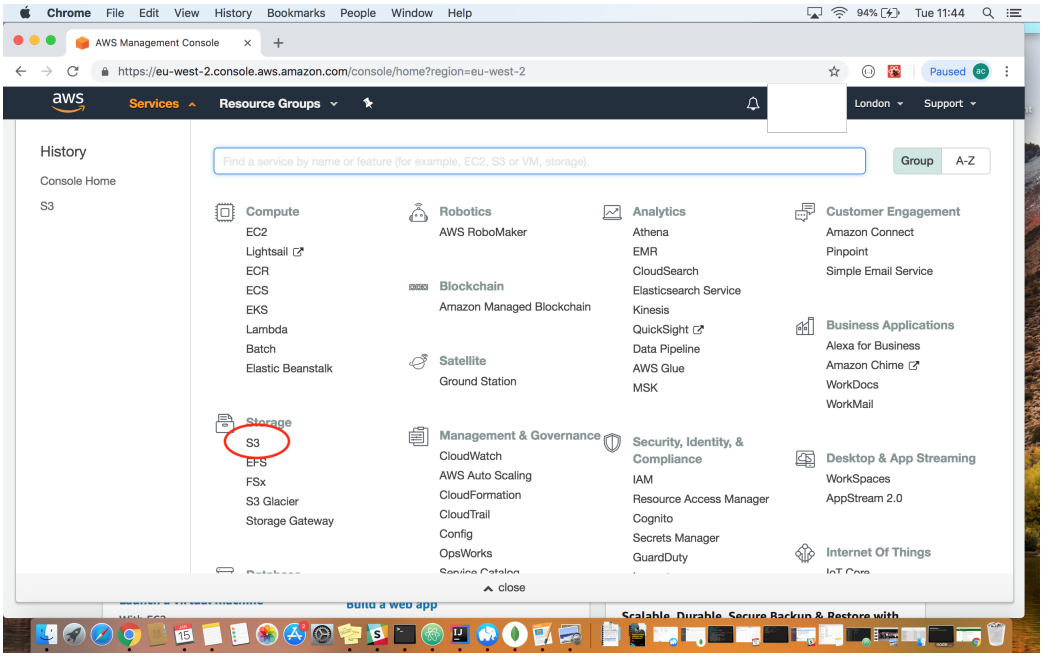
3.1 Introduction

Amazon S3 stands for Amazon Simple Storage Service. It is used to store and retrieve any amount of data, at any time, from anywhere on the web.
It is basically just a folder where put stuff in the cloud. It can be used to host static website i.e a website that does not require server.
The basic unit to store data inside Amazon S3 is the "object". An object consists of the actual data enriched with metadata. The metadata is a set of name-value pairs that provide additional information about the object.

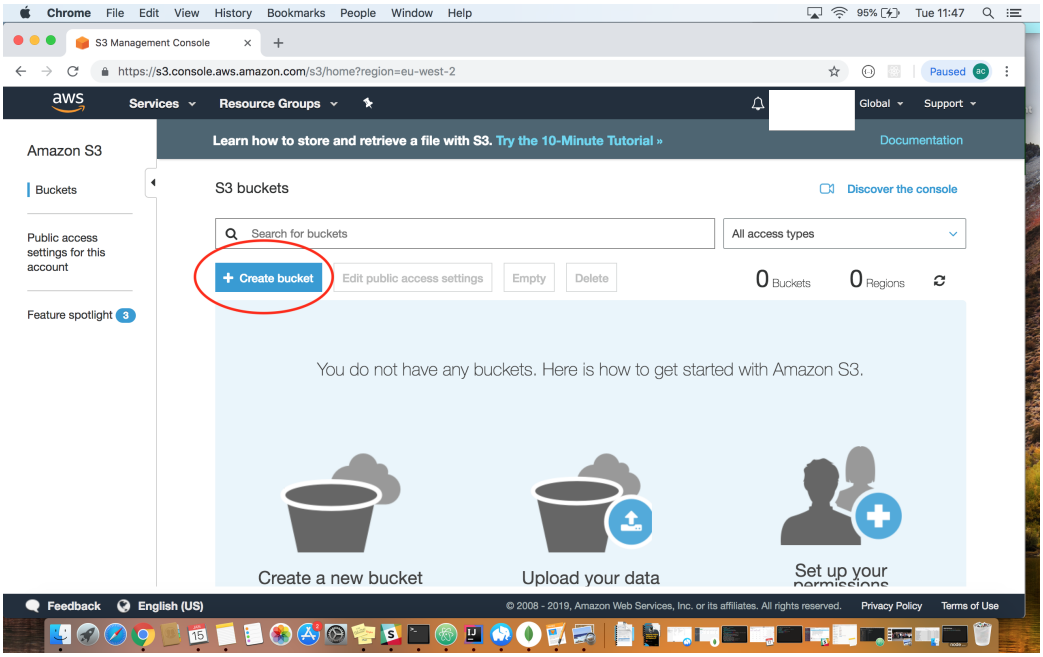
3.2 How to create a S3 Bucket?



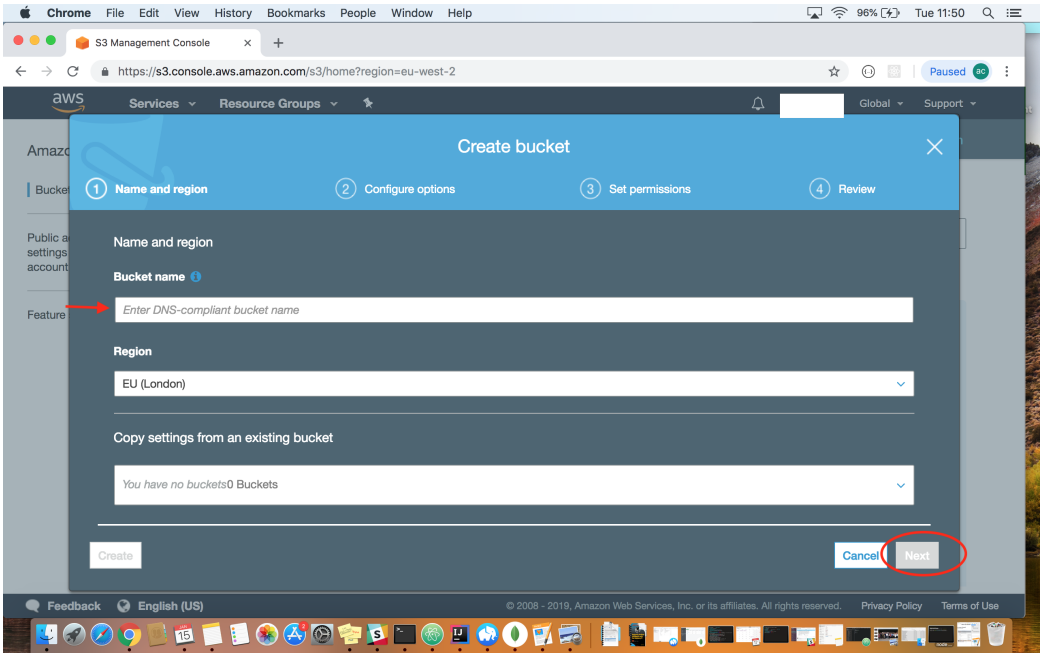
On the top menu click on services



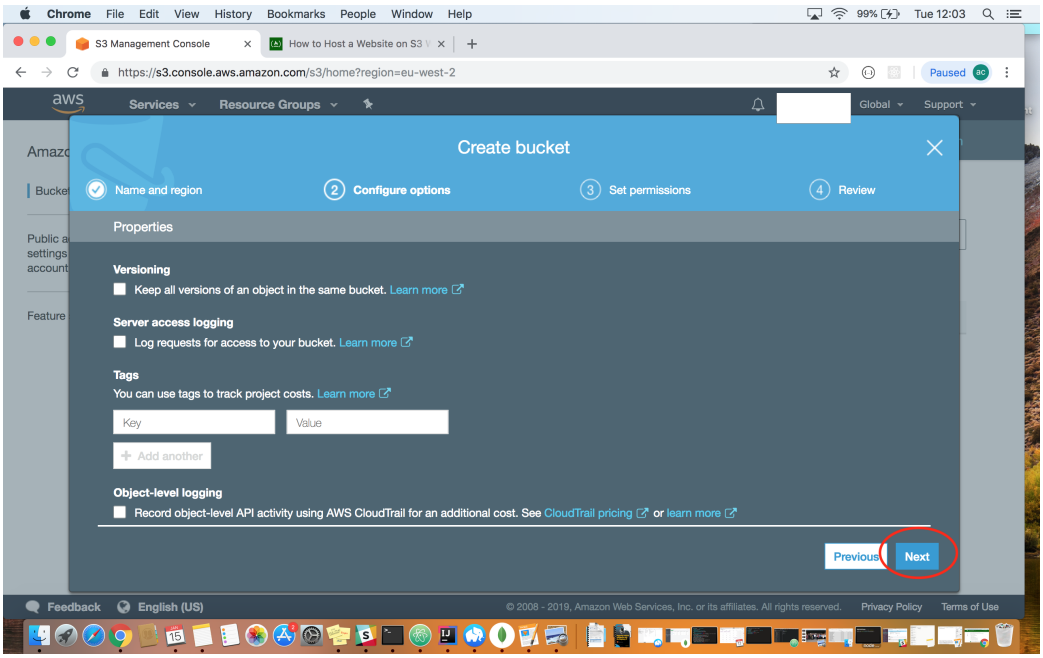
click on S3 underneath Storage



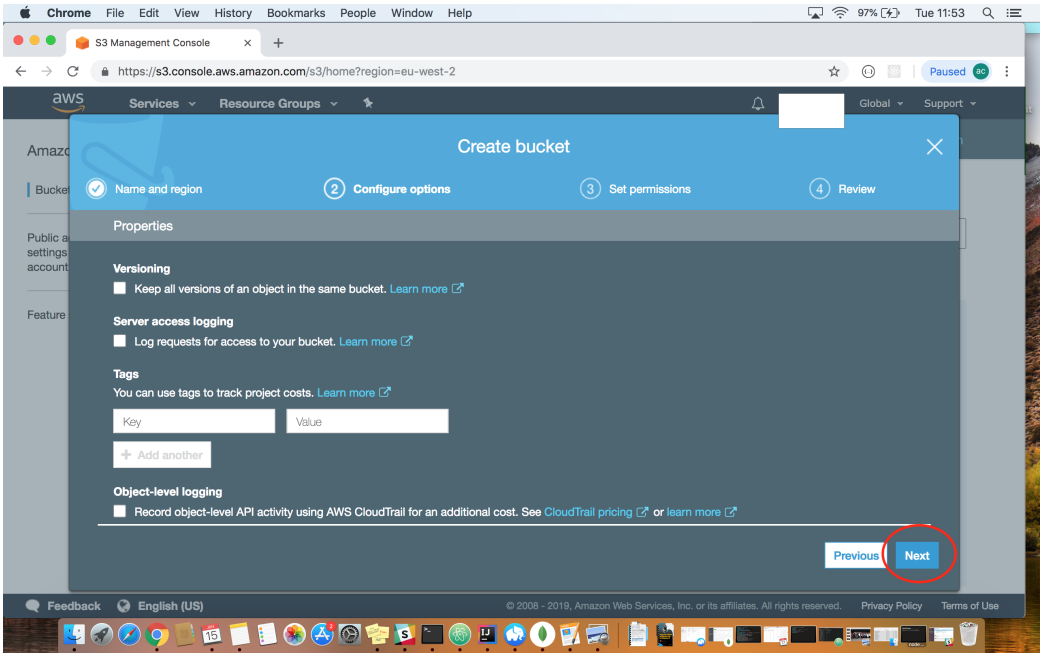
click on Create bucket.



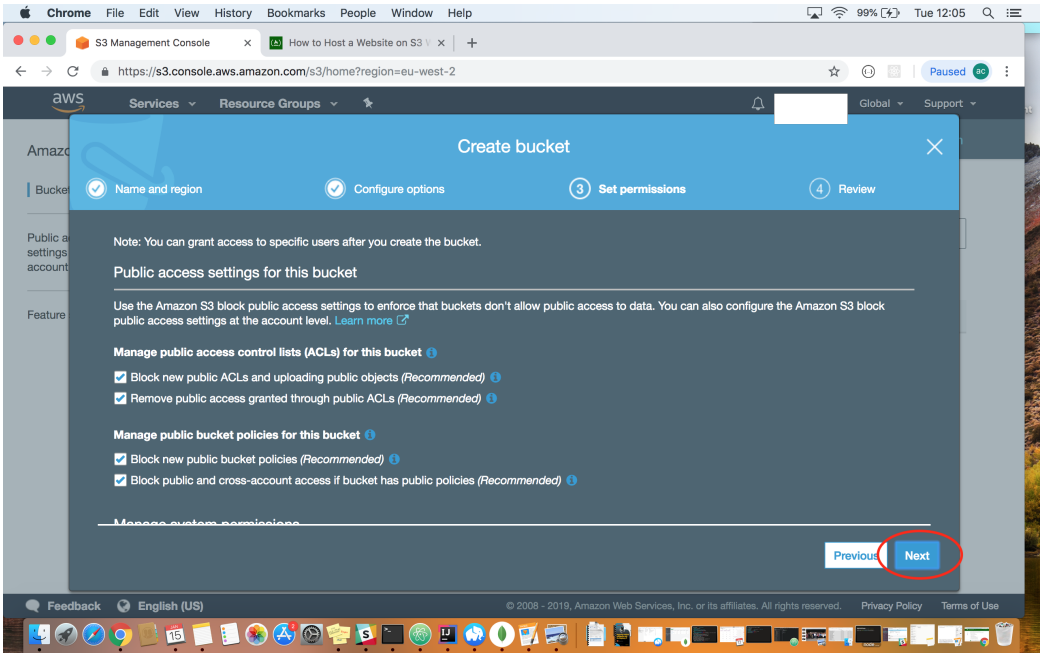
Enter the name of the bucket.
The name of bucket must not contain special character or capital letters.



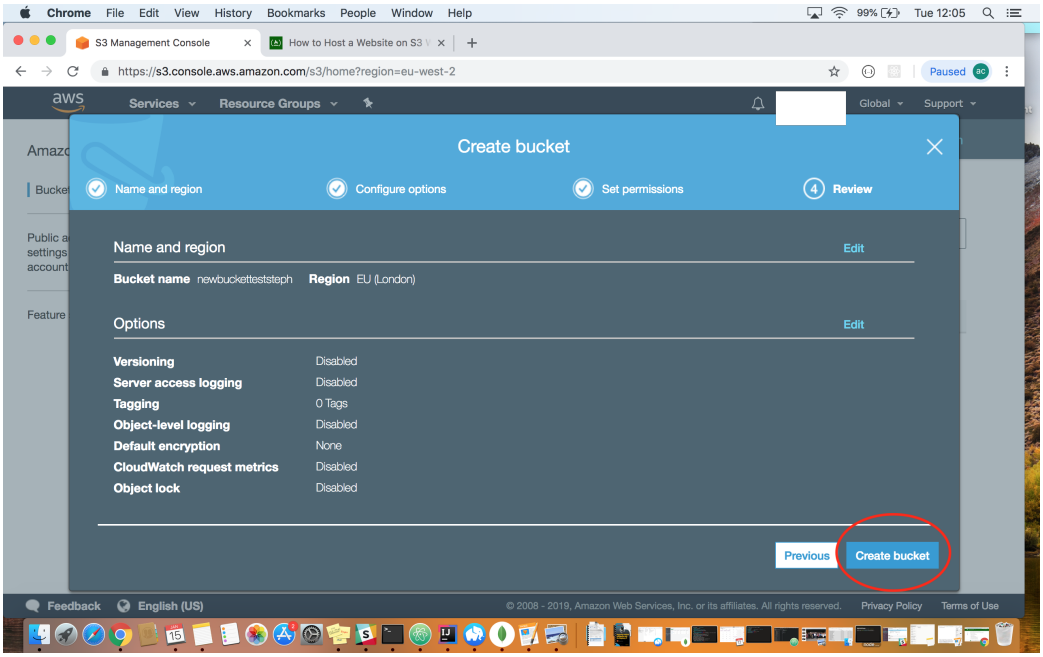
Click on Next and do not change anything



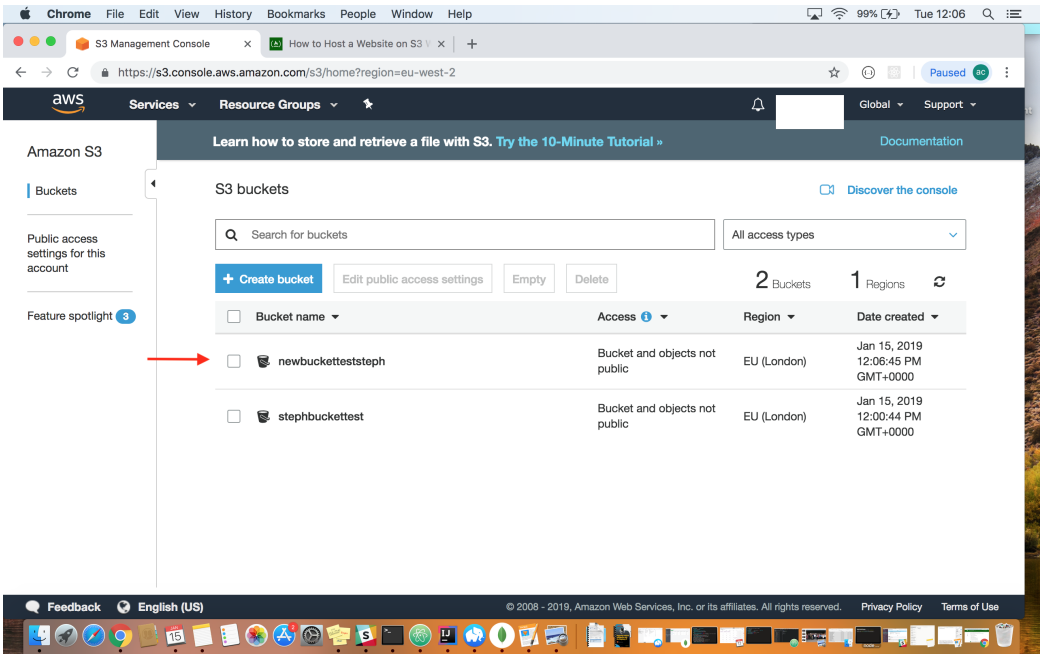
Click on Next



Click on Next



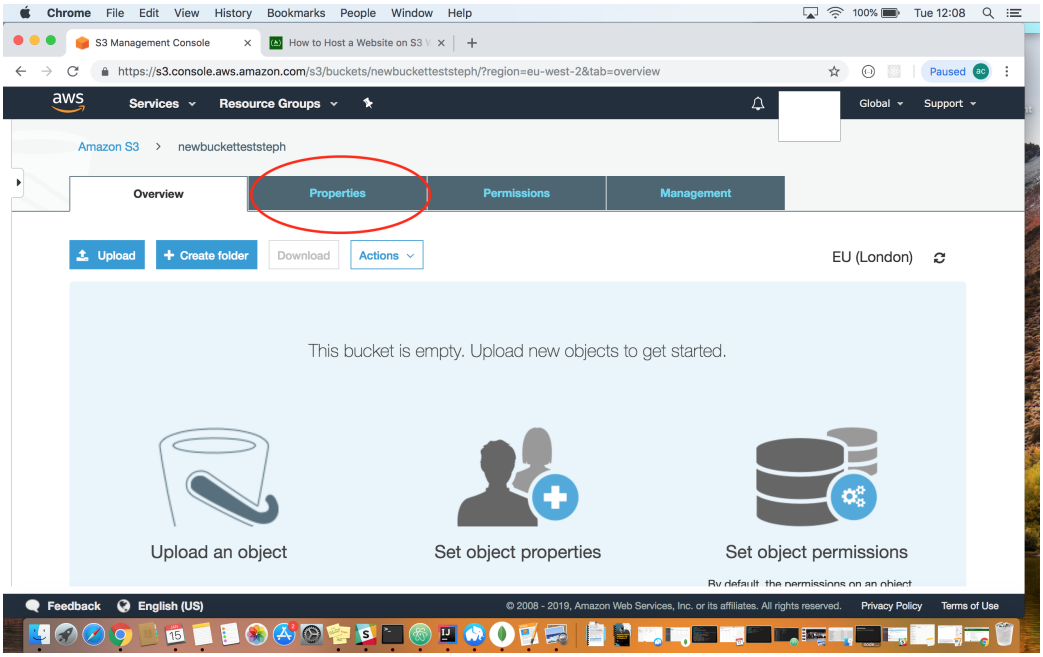
Click on Create bucket



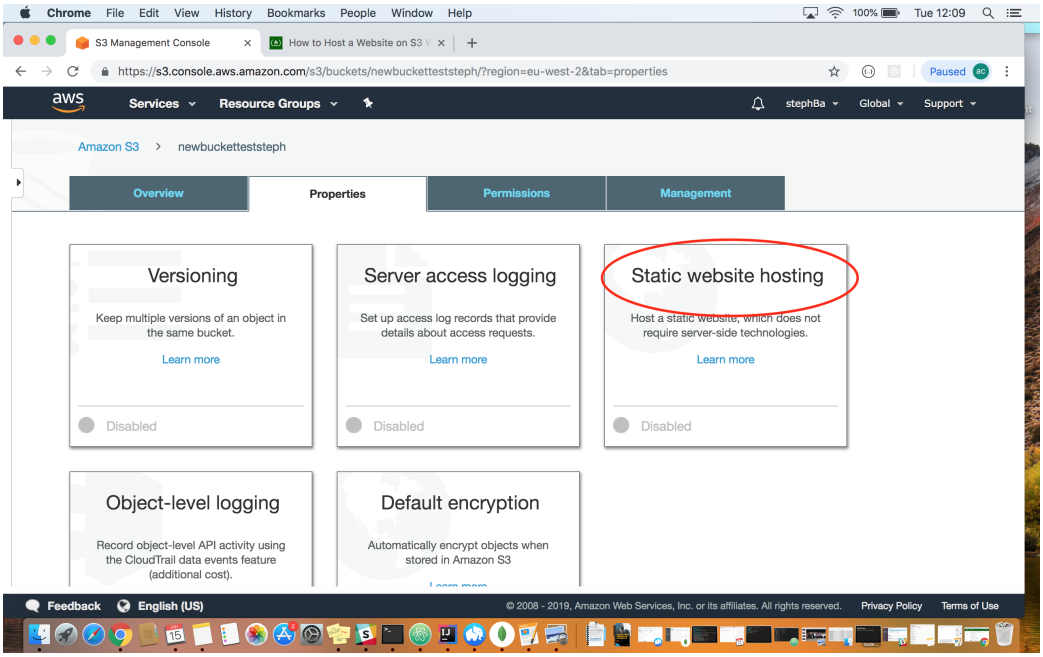
The new bucket will appear on the top of the list.
Click on the name of the bucket in order to access to it. You will be able to manage and change the properties and permissions of the bucket.

3.3 Hosting a static website

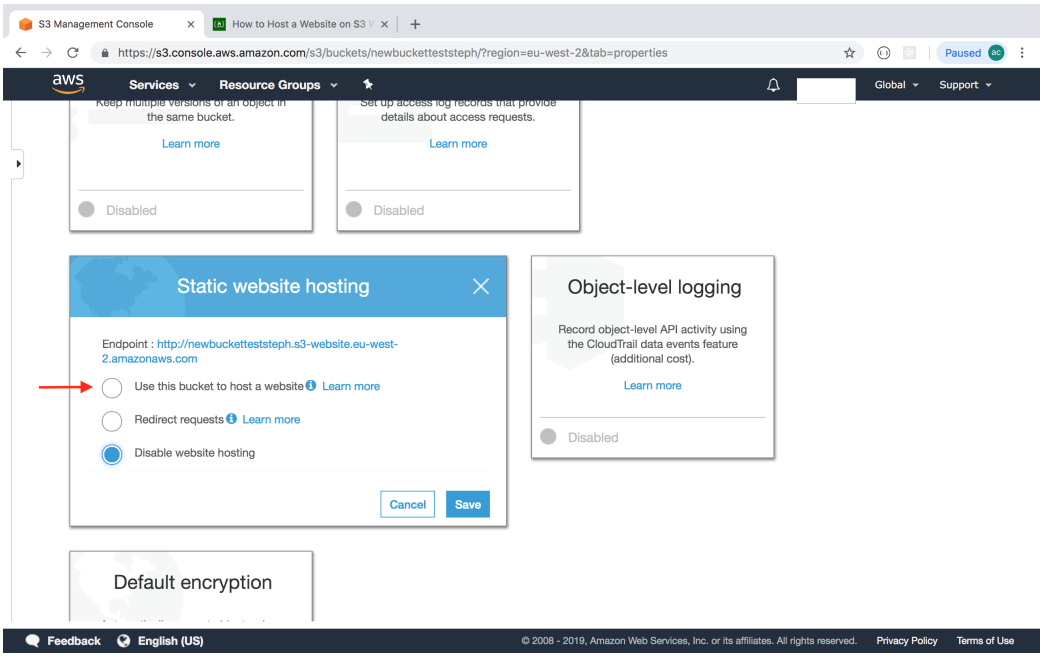
Now we need to enable the S3 bucket to host a static website. In order to do so, follow the steps below:



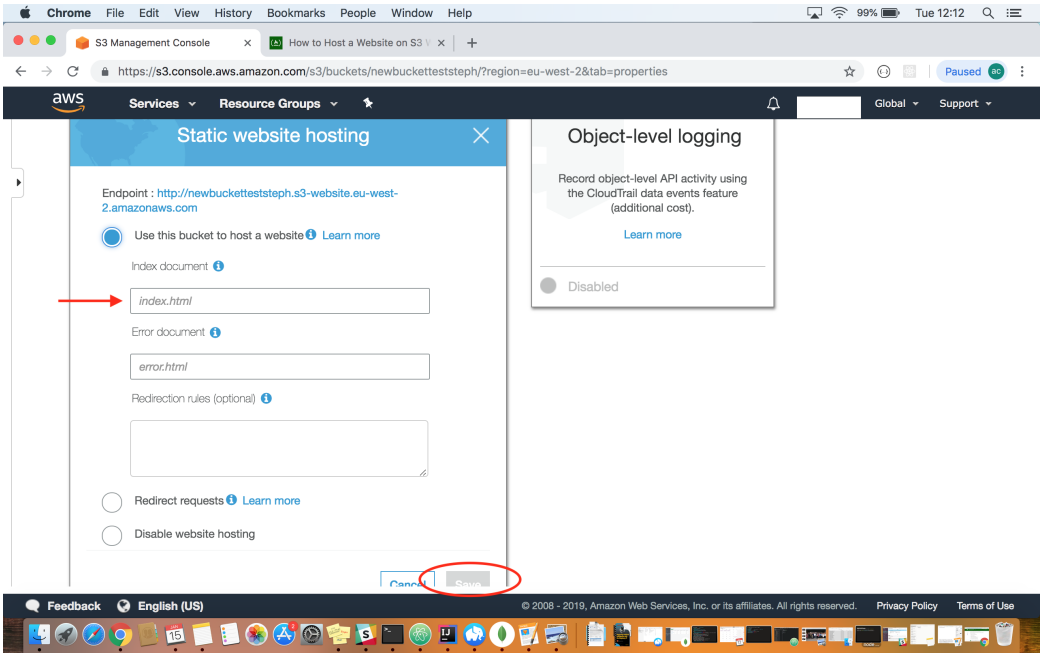
Click on properties



Click on Static website hosting.



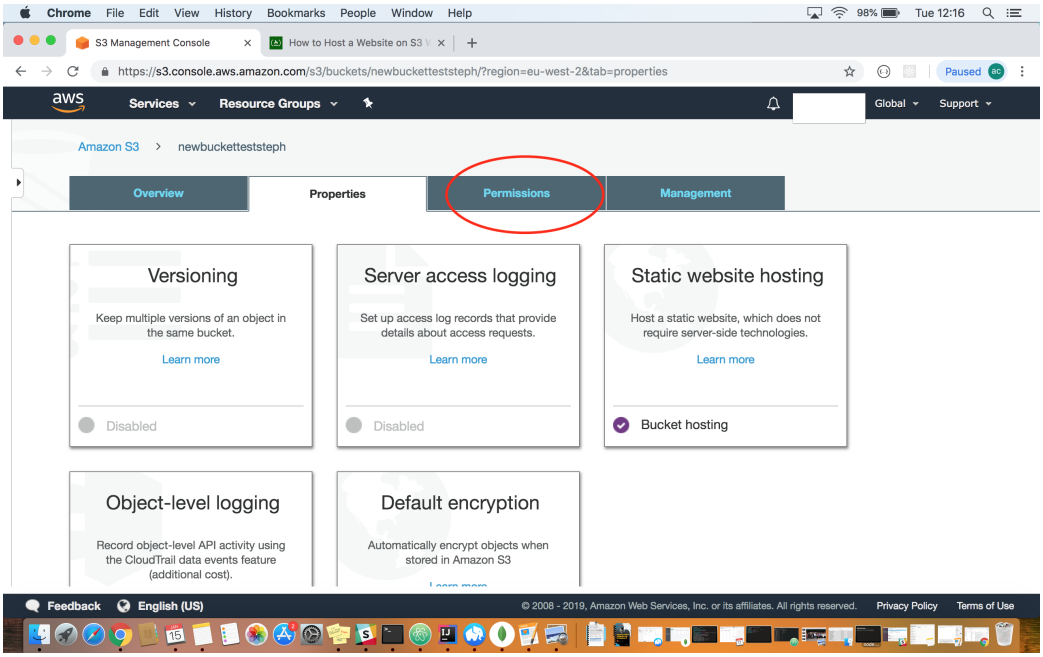
Select Use the bucket t host a website.



In the field Index document, type in index.html and click on save.
The S3 bucket is now ready to receive all file of the website.

3.4 Permissions

We will now modify the permissions for the S3 bucket and see how we can modify them.



Click on Create bucket

3.5 Useful command line to manage S3 buckets

To create a S3 bucket.

```
aws s3 mb s3://nameofthebucket
```

To create a S3 bucket on a specific region.

```
aws s3 mb s3://100daysofdevopsbucket --region name-region-number
```

To list all the S3 buckets.

```
aws s3 mb s3://nameofthebucket
```

3.6 Bucket policy

You can define bucket policy yourself.

Here some example of bucket policies

Granting Permissions to Multiple Accounts with Added Conditions

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AddCannedAcl",
      "Effect":"Allow",
      "Principal": {"AWS": ["arn:aws:iam::1111222233
      33:root","arn:aws:iam::444455556666:root"]},
      "Action":["s3:PutObject","s3:PutObjectAcl"],
      "Resource":["arn:aws:s3:::examplebucket/*"],
      "Condition":{"StringEquals":{"s3:x-amz-acl":
      ["public-read"]}}
    }
  ]
}
```

3.6.1 Specifying ressources

https://docs.aws.amazon.com/AmazonS3/latest/dev/s3-arn-format.html

The following is the common Amazon Resource Name (ARN) format to identify any resources in AWS.

```
arn:partition:service:region:namespace:relative-id
```

For your Amazon S3 resources:

- aws is a common partition name.
- s3 is the service.
- You don't specify Region and namespace.
- For Amazon S3, it can be a bucket-name or a bucket-name/object-key. You can use wild card.

Format for Amazon S3 ressources:

```
arn:aws:s3:::bucket_name
arn:aws:s3:::bucket_name/key_name
```

The wildcard gives access to all object in examplebucket

```
arn:aws:s3:::examplebucket/*
```

3.6.2 Specifying principal

https://docs.aws.amazon.com/AmazonS3/latest/dev/s3-bucket-user-policy-specifying-principal-intro.html

The Principal element specifies the user, account, service, or other entity that is allowed or denied access to a resource.

To grant permissions to an AWS account, identify the account using the following format.

```
"AWS":"account-ARN"
```

For example:

```
"Principal":{"AWS":"arn:aws:iam::AccountNumber-
WithoutHyphens:root"}
```

Example with account ARN.

```
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
```

Example with the account ID

```
"Principal": { "AWS": "123456789012" }
```

You can specify more than one account:

```
"Principal": {
  "AWS": [
    "arn:aws:iam::123456789012:root",
    "999999999999"
  ]
}
```

To give anonymous access, e.g. everybody cann access the bucket, use the wildcard

```
"Principal":"*"

OR

"Principal":{"AWS":"*"}
```

3.6.3 Specifying permissions

https://docs.aws.amazon.com/AmazonS3/latest/dev/using-with-s3-actions.html

You can specify permissions for objects operations. Basically what the user can do on the bucket or the objects. You will define the specific action or actions that will be allowed or denied.

```
"Action": "s3:GetObject"

"Action": ["s3:PutObject","s3:PutObjectAcl"],
```

To grant permission for all Amazon S3 actions.

```
"Action": "s3:*"
```

3.6.4 Specifying Permissions

The access policy language allows you to specify conditions when granting permissions. The Condition element (or Condition block) lets you specify conditions for when a policy is in effect. The Condition element is optional,

3.7 Ressources

4 API Gateway

4.1 Introduction

API Gateway is an AWS service that supports creating, deploying, and managing a RESTful application programming interface to expose backend HTTP endpoints, AWS Lambda functions, and other AWS services.

An API Gateway API is a collection of resources and methods that can be integrated with Lambda functions, other AWS services, or HTTP endpoints in the backend. The API consists of resources that form the API structure. Each API resource can expose one or more API methods that must have unique HTTP verbs.

4.2 Ressources

5 Lambda functions

5.1 Introduction

5.2 Ressources

6 DynamoDB

6.1 Introduction

In DynamoDB, tables, items, and attributes are the core components that you work with.

A **table** is a collection of **items**, and each item is a collection of **attributes**.

DynamoDB uses primary keys to uniquely identify each item in a table.

6.1.1 Tables, Items, Attributes

The following are the basic DynamoDB components:

- **Tables** – DynamoDB stores data in tables. A table is a collection of data.
  For example, see the example table called People that you could use to store personal contact information about friends, family, or anyone else of interest. You could also have a Cars table to store information about vehicles that people drive.

- **Items** – An item is a group of attributes that is uniquely identifiable among all of the other items. Each table contains zero or more items.
  For example, in a People table, each item represents a person. For a Cars table, each item represents one vehicle. Items in DynamoDB are similar in many ways to rows, records, or tuples in other database systems. In DynamoDB, there is no limit to the number of items you can store in a table.

- **Attributes** – An attribute is a fundamental data element, something that does not need to be broken down any further. Each item is composed of one or more attributes.
  For example, an item in a People table contains attributes called PersonID, LastName, FirstName, and so on. For a Department table, an item might have attributes such as DepartmentID, Name, Manager, and so on. Attributes in DynamoDB are similar in many ways to fields or columns in other database systems.

Each item in the table has a unique identifier, or primary key, that distinguishes the item from all of the others in the table.

6.1.2 Primary Keys

When you create a table, in addition to the table name, you must specify the primary key of the table. The primary key uniquely identifies each item in the table, so that no two items can have the same key.

DynamoDB supports two different kinds of primary keys:

- **Partition key** – A simple primary key, composed of one attribute known as the partition key. DynamoDB uses the partition key's value as input to an internal hash function. The output from the hash function determines the partition (physical storage internal to DynamoDB) in which the item will be stored. In a table that has only a partition key, no two items can have the same partition key value. The People table described in Tables, Items, and Attributes is an example of a table with a simple primary key (PersonID). You can access any item in the People table directly by providing the PersonId value for that item.

- **Partition key and sort key** – Referred to as a composite primary key, this type of key is composed of two attributes. The first attribute is the partition key, and the second attribute is the sort key. DynamoDB uses the partition key value as input to an internal hash function. The output from the hash function determines the partition (physical storage internal to DynamoDB) in which the item will be stored. All items with the same partition key value are stored together, in sorted order by sort key value. In a table that has a partition key and a sort key, it's possible for two items to have the same partition key value. However, those two items must have different sort key values. The Music table described in Tables, Items, and Attributes is an example of a table with a composite primary key (Artist and SongTitle). You can access any item in the Music table directly, if you provide the Artist and SongTitlevalues for that item. A composite primary key gives you additional flexibility when querying data. For example, if you provide only the value for Artist, DynamoDB retrieves all of the songs by that artist. To retrieve only a subset of songs by a particular artist, you can provide a value for Artist along with a range of values for SongTitle.

6.2 How to install DynamoDB locally?

Create a folder where you will install DynamoDB.

Download the zip file from **the website Amazon Web Services**

Unzip the zip file.

This code will run the server of DynamoDB.

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

You will need to configure DynamoDB with aws configure by adding your keys, the region and the output format:

```
AWS Access Key ID [None]: type your Access Key
AWS Secret Access Key [None]: type your Secret
Access Key
Default region name [None]: type your region
Default output format [None]: json
```

6.3 How to create a table?

6.3.1 From the AWS CLI

The command line to create a table in the database is

```
aws dynamodb create-table
```

However, you will need to define the following information in order to create a table:

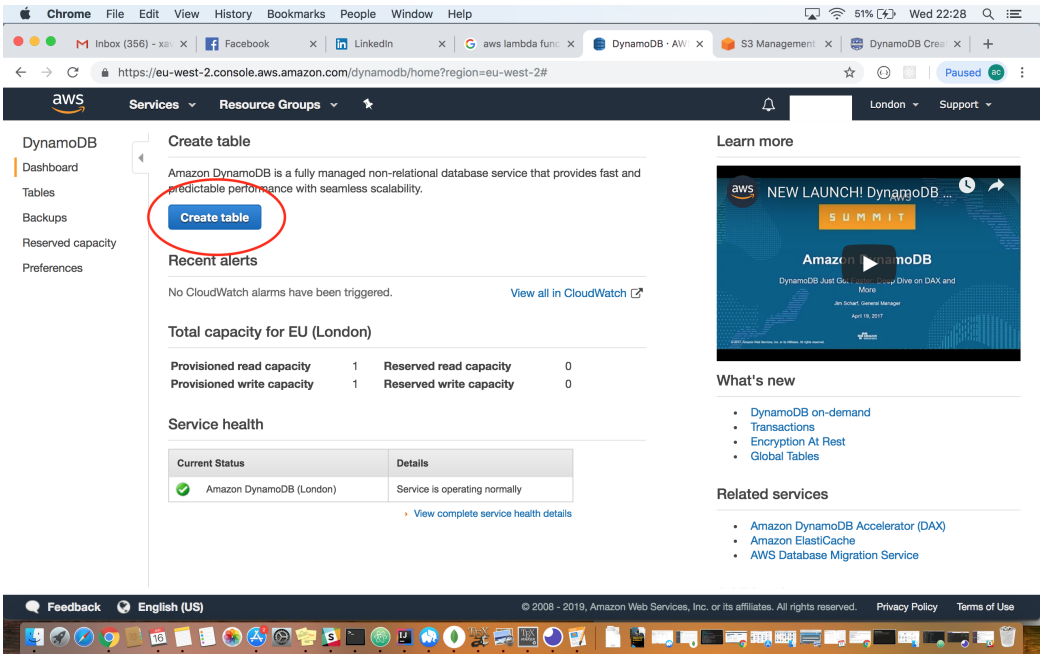
- –attribute-definitions
- –key-schema
- –provisioned-throughput
- –table-name

Example of code:

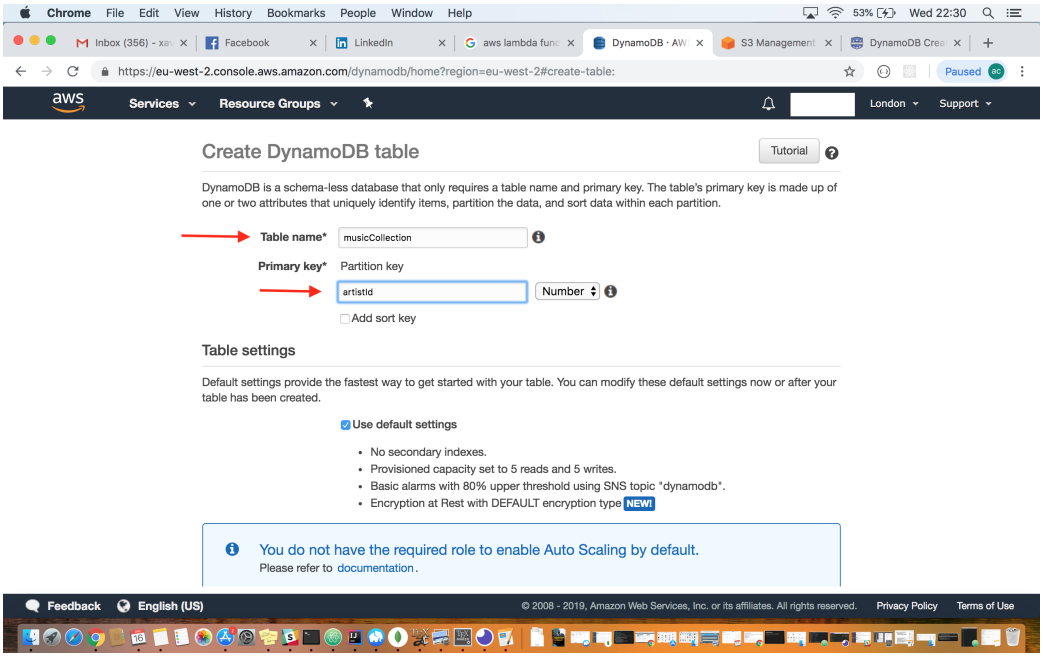
```
aws dynamodb create-table --table-name Client \
--attribute-definitions AttributeName=client_id,
AttributeType=S \
--key-schema AttributeName=client_id,KeyType=HASH \
--provisioned-throughput ReadCapacityUnits=1,
WriteCapacityUnits=1
```

N.B: you can use \ to go to the next line. It will make the clearer.

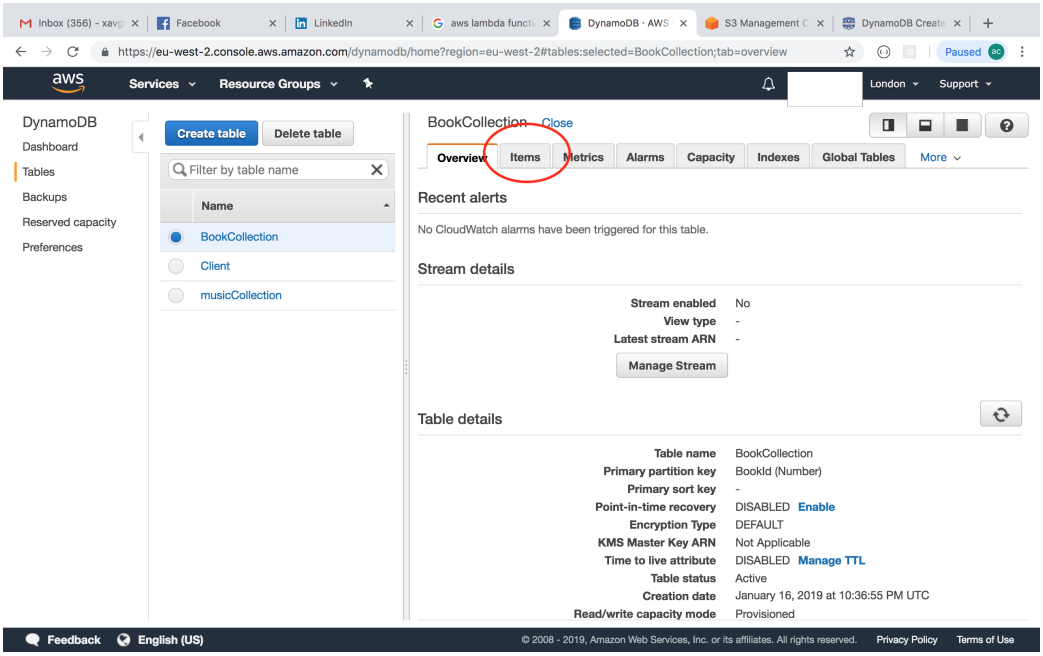
6.3.2 From the AWS Console



Click on Create Table

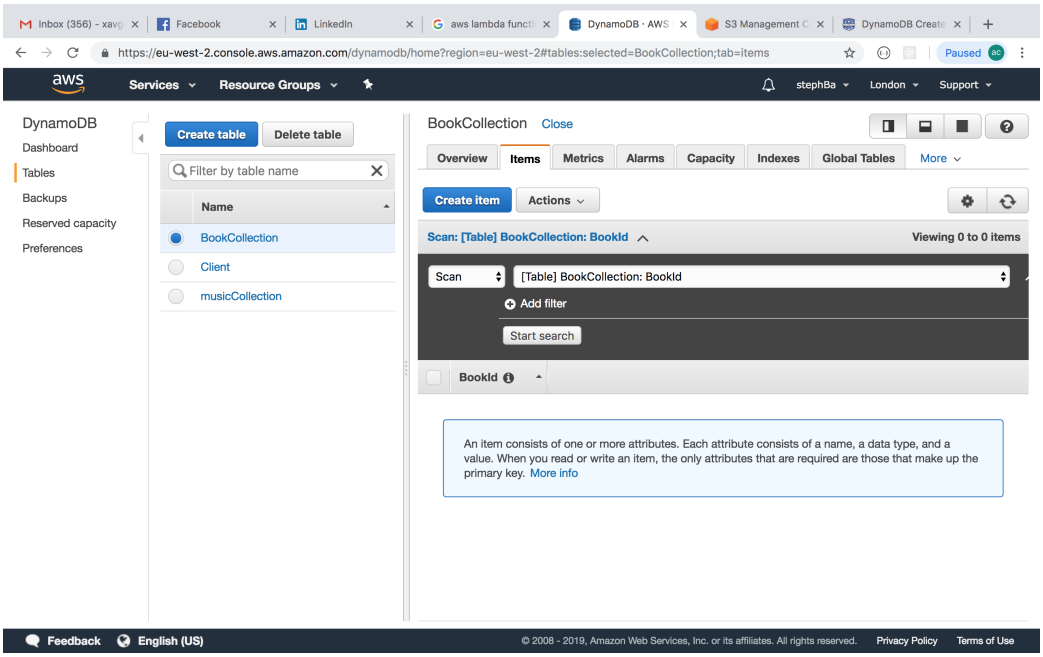


Enter the name of the table
Enter the name of the Partition key, that will be used for the primary key, and the data type of the partition key and click on Create.

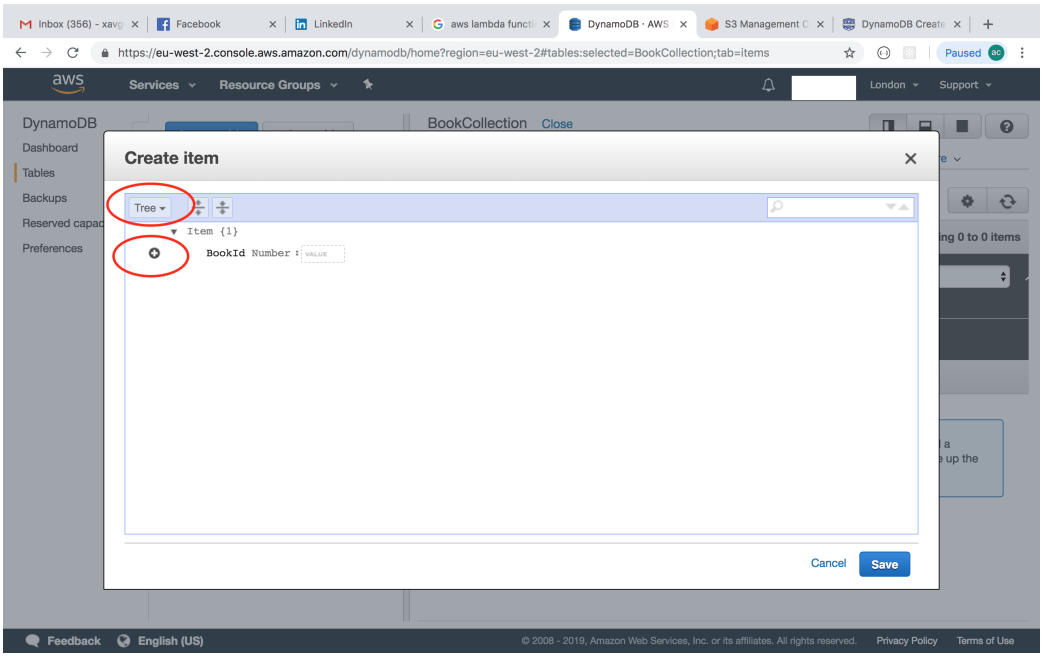


You will be redirected to a page that shows the list of your tables and an

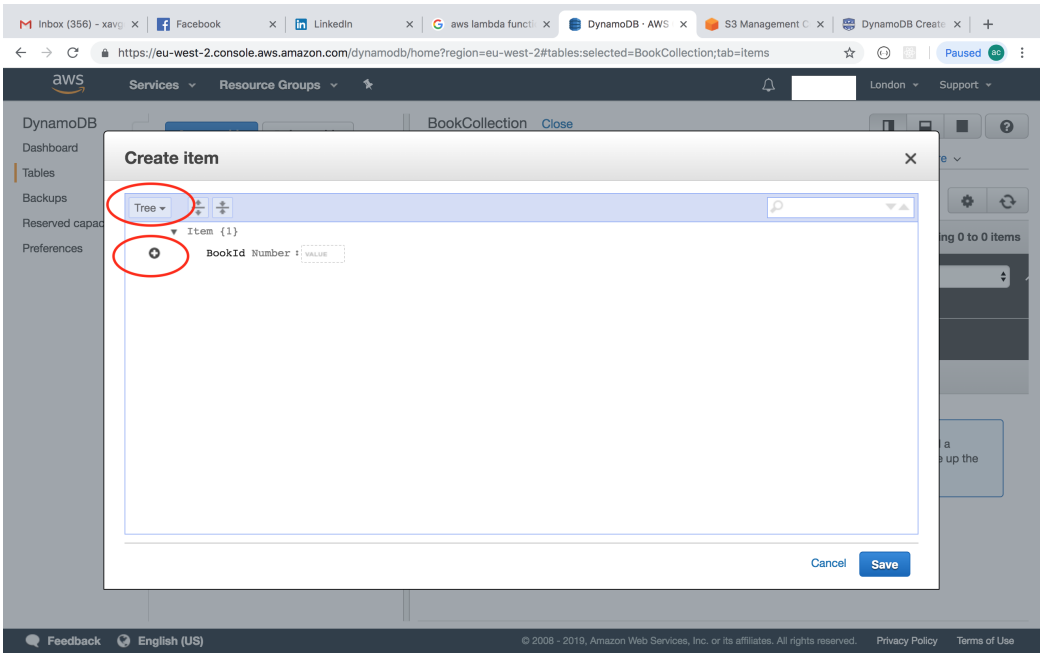
overview of the table that you just created. Click on items in order to create the items of the table.



Click on Create item to create an item and add it to the table.
A new window will pop up.



You can change th eview of the item by clicking on tree. Two view tree format like on the picture or json format.
click on the + sign to add another attribute.



You can choose between Append, Select or Remove.

6.4 How to get an item form the DynamoDB?

6.5 How to put an item form the DynamoDB?

6.6 Ressources