Homework 2

1. Create a static website using AWS S3 and explain in great technical detail what you did.

You need to have the AWS CLI already installed (this is explained in the previous homework).

1. Create a bucket

```
aws s3api create-bucket --bucket <nameofsubdomain.domain.com> --region
us-east-1
```

This creates a new S3 bucket. (you must already have your AWS Access Key ID created and authorized). For the bucket name it must be something unique, otherwise it would create a conflict with other buckets (naming rules). Then the region must be specified, in this case we'll use the us-east-1.

2. We need to have a **bucket_policy.json** file in order to grant public read access to the bucket, this way anyone in the internet who reaches your bucket will be able to do so; this file needs to have the following parameters:

- The Version date you need.
- Sid: es el statment id (esto da el permiso).
- Effect: en allow o deny.
- Principal: que por el * significa que es todos los users.
- Action: s3:GetObject (para que agarre nuestro objeto)
- Resource: (el recurso, bucket name)

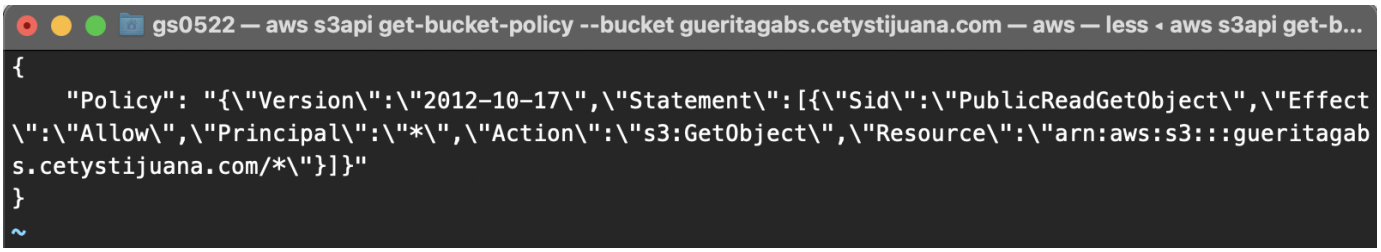
```
{"Version": "2012-10-17",
"Statement": [
    {
        "Sid": "PublicReadGetObject",
        "Effect": "Allow",
        "Principal": "*",
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::<bucket_name>/*"
    }
]
}
```

The command to upload this file into the bucket is: (you need to be in the path where the .JSON file is located) you use the *put-bucket-policy* to indicate you are putting something into the bucket, then you specify *the bucket name* and the *path file* of your document.

```
aws s3api put-bucket-policy --bucket <bucket_name> --policy
file:///<path>
```

After a couple of seconds, you may check that the file was uploaded correctly.

```
 aws s3api get-bucket-policy --bucket <bucket_name>>
```



This what it should look like once the file is uploaded correctly. If you want to check if the page exists as a subdomain you may run the next command:

```
aws s3 website s3://<bucket_name>
```

3. To make the bucket host an static site, we need to make the bucket able to have the parameters of website, for that we need the *index.html* an d the *error.html*, in order to make this we need this command:

```
aws s3 website s3://<bucket_name> --index-document index.html --error-document error.html
```

Now, this file doesn't exits yet we need to create it, we can either do that in the finder or through a command, for that we can use the next command, first to create a directory.

```
mkdir <name_of_the_directory>
```

Then we need to move to that directory

```
cd <name_of_the_directory>
```

Once in that file we can create the documents.

```
touch index.html  // touch error.html
```

With those files created we can add an example text such as “Hello World” for the index file and “Oops, something is wrong 😮 “ for the error file. Once that is done we need to push those documents to the bucket, to do that we use the next command:

(the order is **origin and destiny** in this case the . is the origin and then the bucket name the destiny)

```
aws s3 sync . s3://<bucket_name>
```

Finally to verify that the files where pushed correctly we can use the next command:

```
aws s3api get get-bucket-website --bucket <bucket_name>
```

2. Link your website to a subdomain of cetystijuana.com and explain in great technical detail what you did

In order to get a subdomain on the main domain page you need to use route53 de DNS (Domain name system) to do so we need a JSON file.

The `UPSERT` part means that if it already exists a resource on Route 53, it will update it with the values in the request.

The `Name` part means the bucket name (with the subdomain)

For `Type` we use CNAME (Canonical Name) is a type of DNS record that maps an alias name to a true domain name.

The `TTL` (Time to Live) is set to 600 so that means how long the CDN will hold onto your data before it retrieves updated information.

The `ResourceRecords` This has the complete path of the s3 website.

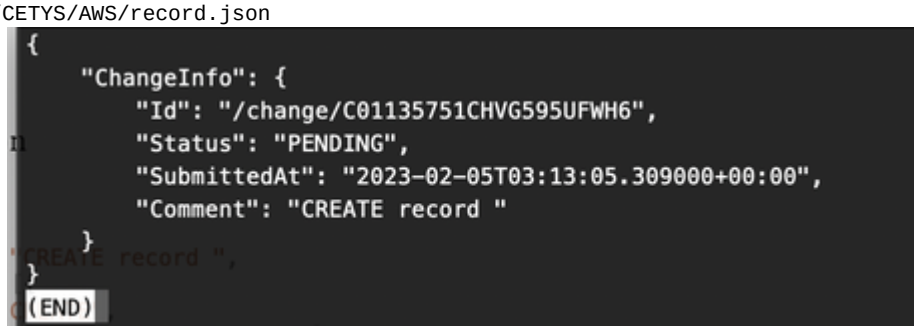
```
{
  "Comment": "CREATE record ",
  "Changes": [{
  "Action": "UPSERT",
             "ResourceRecordSet": {
                         "Name": "gueritagabs.cetystijuana.com",
                         "Type": "CNAME",
                         "TTL": 600,
                       "ResourceRecords": [{ "Value":  "gueritagabs.
cetystijuana.com.s3-website-us-east-1.amazonaws.com"}]
}}]
}
```

Once we have thaat file, we need to uploaded, to do so in the terminal we need to be in the correct path where our JSON is stored, then we can run the next command line

`aws route53 change-resource-record-sets --hosted-zone-id Z03346142C3RKH191036Y --change-batch file://<path_of_the_JSON_file>`

For example, mine looked like this:

`aws route53 change-resource-record-sets --hosted-zone-id Z03346142C3RKH191036Y --change-batch file:///Users/gs0522/Documents/CETYS/AWS/record.json`



And if we want to check once the STATUS is completed, we need to take that generated ID Number and run in the next command:

`aws route53 list-resource-record-sets --hosted-zone-id <id_number>`

to retrieve the next status as **INSYNC**, meaning that everything is up and running.

This command is needed in order to do any type of modification (create, change, delete) for our subdomain route from my bucket

3. Add yourself to the list of students in Cetystijuana.com.

This a vital step, without this anyone who doesn't know what the specific sub-route is for another part of the website, won't be able to access it.

The page has a section that says Students, in here we first need to download all the information/documents that are up in the webpage so far, for that we use:

```
aws s3 sync s3://cetystijuana.com .
```

Meaning that I am bringing **everything** from the webpage to my local computer.

Then In the asstes/img/team folder I uploaded my picture.

And on the index.html file I added my division for the student section.

Once that is done I need to upload all my local changed to the website. (If someone was editing his part while I do this section of the homework they would loose their changes if they don't pull the latest update)

```
aws s3 sync . s3://cetystijuana.com
```

In the part inside my profile where it says **Gaby's Website** is a hyperlink that will get you to my sub-page.

For that I just need to update my index.html file with the format that I want my webpage to be displayed and push my changes with the command:

```
aws s3 sync . s3://gueritagabs.cetystijuana.com
```

4. Create an entry in the Students DynamoDB table using the cli with the following model:

To insert an entry into the Students DynamoDB we need to use a command where it specifies that we are going to insert something with *put-item* in an specific table called *Students*, accepting the next parameters:

- ID: 033229
- Full Name: Gabriela Sanchez Alcaraz
- Personal Website: gueritagabs.cetystijuana.com

```
aws dynamodb put-item --table-name Students --item '{"id": {"S": "MATRICULA"}, "full_name": {"S": "FULL NAME"}, "personal_website": {"S": "Bucker name / personal-website"}}'
```

And to confirm that it worked we can use the `aws dynamodb scan --table-name Students` command to check the table of Students.

5. Watch the Not Just Code Monkeys by Martin Fowler and write your comments in your personal static website.

Something Fowler says right at the begging of the video is how, yes Agile makes the development work ready not until the documentation is ready, but more like an in-between. The problem here is that most of the times the *stories* are created not by developers, and this can create a sort of “tension” between the analyst and the developers, this because we developers can’t really do whatever we want that is going to make the program better, any suggestion needs to go through certain processes in order to be taken in consideration. This should be a win-win situation where a conversation is a MUST in order to make sure all ideas are taken in consideration for the final client to be happy with their product.

Now in this career there is a lot of things we need to learn, but we don't have to learn everything and being mediocre at that, its better if we focus in one thing but be great at it.

Another topic he touches at the talk is about being an ethic programmer, if we are not true to our moral values, we can trick the users into stuff that will make them bad. Now we need to keep in mind that we programmers need to have our back, and think about the bigger picture, how we may impact either our programmer colleagues or the users (promote women, do not create differential groups); and if we know a problem exists, we need to fight that, not just ignore it.