# Assignment 3

**1. Create a CloudFront distribution for your website and explain each step with great technical detail. 50 points.**

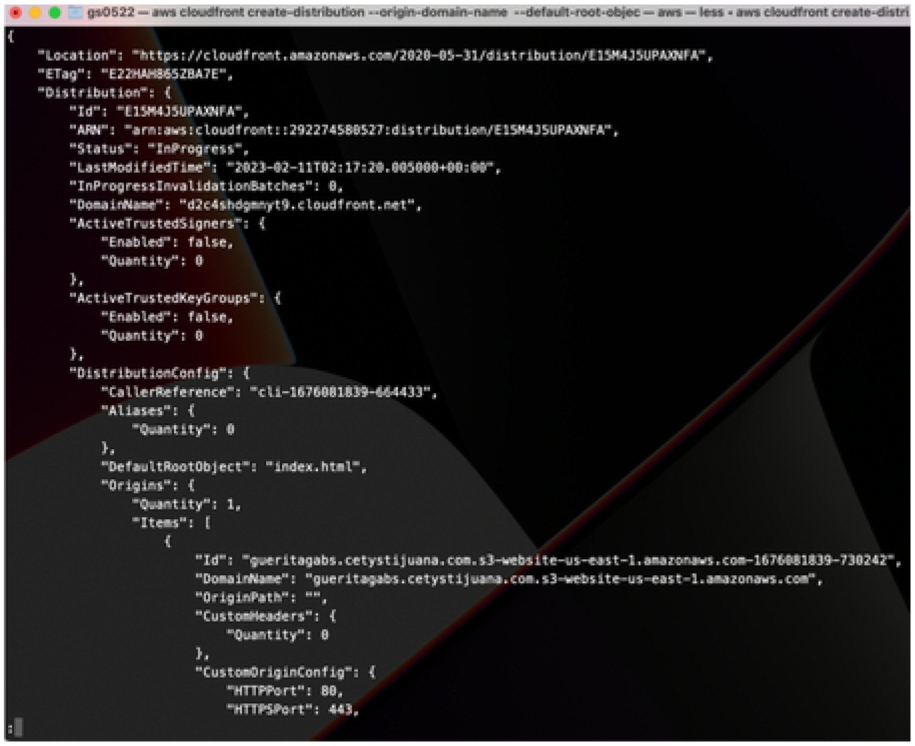
CloudFront is a CDN (Content Delivery Network). The CloudFront retrieves data from the Amazon S3 bucket and distributes it to multiple data center locations. The data that delivers through a network of data centers is called edge locations.

The nearest edge location is routed when the user requests for data, so we will get some benefits like the lowest latency, low network traffic, fast access to data, etc.

The first step is to create the CloudFront distribution, this will help that when someone tries to get the website from London, it takes the information from a nearby edge location, and not from California. In other words, it will gather the information from the edge location with the lowest latency. The next command will create the distribution:

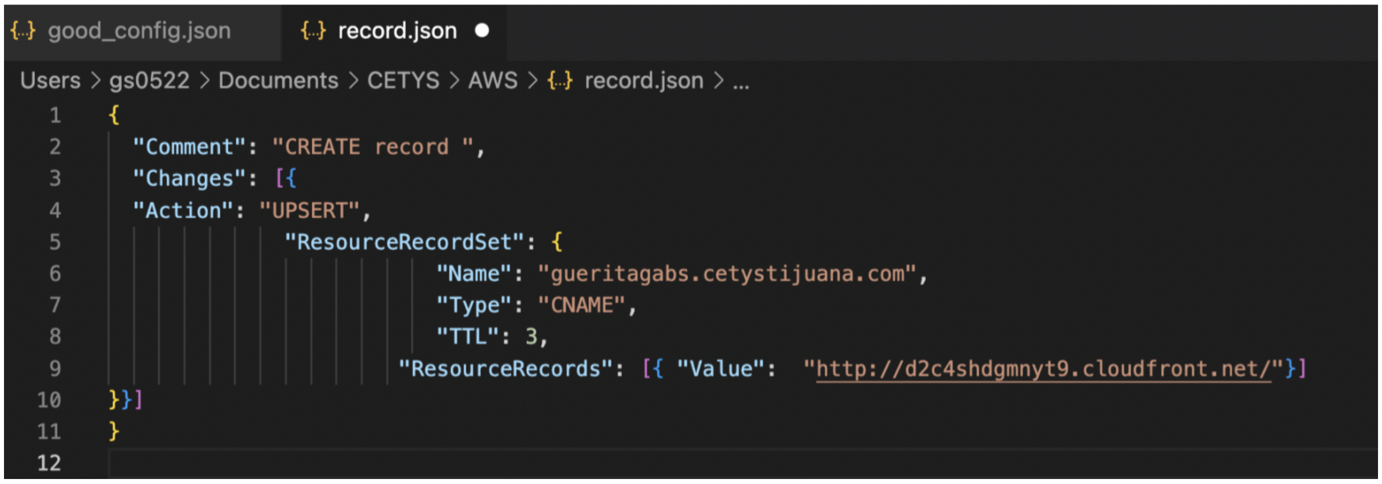
```
aws cloudfront create-distribution --origin-domain-name gueritagabs.
cetystijuana.coms3.amazonaws.com --default-root-object index.html
```

What this command does is that it *creates a distribution* on the origin domain name that points out to the s3 bucket, in my case, it's *gueritagabs.cetystijuana.com*, after that, we specify what record we want, in my case, it's the *index.html*. After we run the command, something similar should be displayed:


gs0522 — aws cloudfront create-distribution --origin-domain-name --default-root-objec — aws — less ‹ aws cloudfront create-distri

```
{
    "Location": "https://cloudfront.amazonaws.com/2020-05-31/distribution/E15M4J5UPAXNFA",
    "ETag": "E22HAH865ZBA7E",
    "Distribution": {
        "Id": "E15M4J5UPAXNFA",
        "ARN": "arn:aws:cloudfront::292274580527:distribution/E15M4J5UPAXNFA",
        "Status": "InProgress",
        "LastModifiedTime": "2023-02-11T02:17:20.005000+00:00",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d2c4shdgmnyt9.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "ActiveTrustedKeyGroups": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-1676081839-664433",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "gueritagabs.cetystijuana.com.s3-website-us-east-1.amazonaws.com-1676081839-730242",
                        "DomainName": "gueritagabs.cetystijuana.com.s3-website-us-east-1.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "CustomOriginConfig": {
                            "HTTPPort": 80,
                            "HTTPSPort": 443,
:
```

Here we can see that our domain name distribution is *d2c4shdgmnyt9.cloudfront.net* which if someone else tries to get to it, it won't be "user friendly" since they will be searching for "gueritagabs.cetystijuana.com", in order to have that pointing to our personalized domain name, we need to change the **record.json** archive.



good_config.json    record.json

Users > gs0522 > Documents > CETYS > AWS > record.json > ...

```json
{
  "Comment": "CREATE record ",
  "Changes": [{
  "Action": "UPSERT",
              "ResourceRecordSet": {
                          "Name": "gueritagabs.cetystijuana.com",
                          "Type": "CNAME",
                          "TTL": 3,
                       "ResourceRecords": [{ "Value":  "http://d2c4shdgmnyt9.cloudfront.net/"}]
}}]
}
```

As you can see the **Action: UPSERT** indicates that we are going to insert and update the following information values, the **name** (how it will be searched by), **Type: CNAME** which is an alternate domain name, it lets you use your own domain name instead of using the domain name that CloudFront assigns automatically, **TTL:3** is the resource record cache time to live, how long it will hold on to your data (cache) before it gets updated; before it was on 600 and we changed it to 3 seconds, that way the changes will be shown on the shortest amount of time, **ResourceRec ords:** it needs the domain name created by CloudFront which was shown on the step before, in my case it is d2c4shdgmnyt9.cloudfront.net.

Now we need to add those changes with the next command:

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

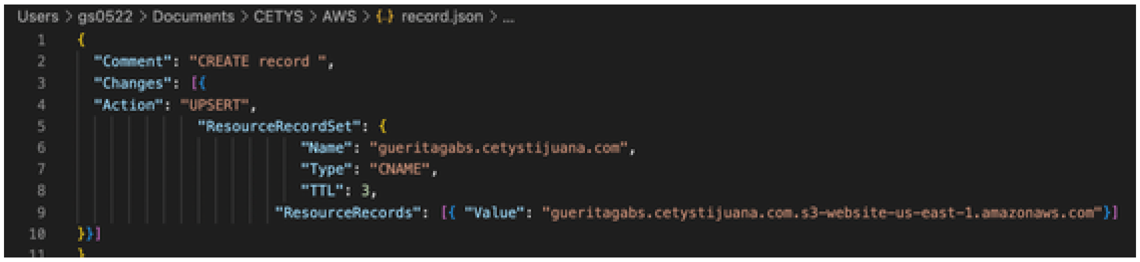
Here we are using route53 to validate the changes in the request and then either make all or none of the changes in the change batch request where we send the parameter of the hosted id zone (the hosted zone is the container for DNS records), then we send the instruction to change the selected file, in this case, is the record.json.

**2. Update your DNS Record to route to the CloudFront endpoint and explain each step with great technical detail. 10 points.**

Now that we have created our CloudFront distribution, we need to update our DNR record to be pointing to the new domain name, here we use a CNAME, which is known for using an alternate domain name that points to our webpage, instead of having to put the bucket name; this will also help with latency since they will be sent to the closest edge location to them.

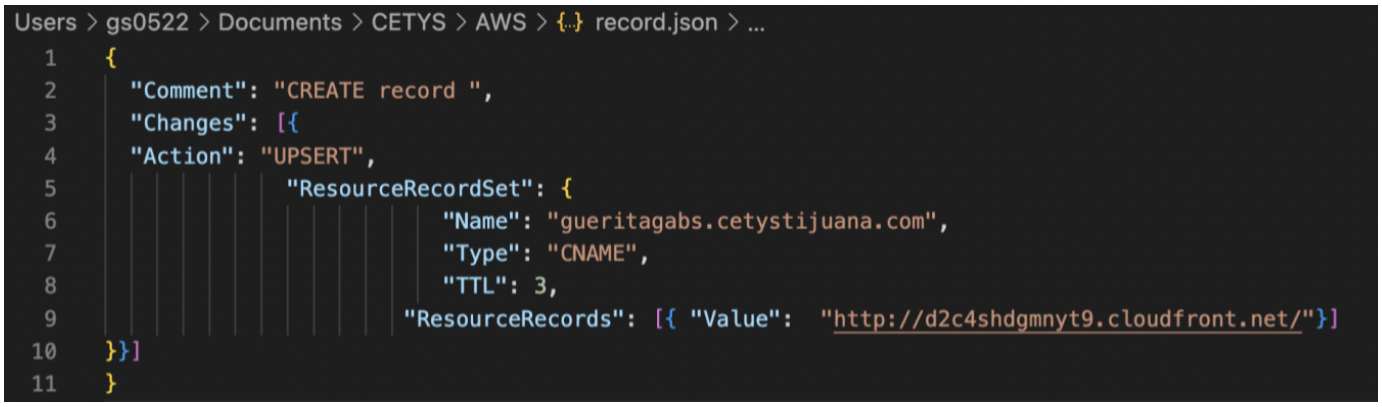
We need to change our **record.json** file; I'm

going to show the before and after:



Users > gs0522 > Documents > CETYS > AWS > record.json > ...

```json
{
  "Comment": "CREATE record ",
  "Changes": [{
  "Action": "UPSERT",
              "ResourceRecordSet": {
                          "Name": "gueritagabs.cetystijuana.com",
                          "Type": "CNAME",
                          "TTL": 3,
                       "ResourceRecords": [{ "Value": "gueritagabs.cetystijuana.com.s3-website-us-east-1.amazonaws.com"}]
}}]
}
```



```
Users > gs0522 > Documents > CETYS > AWS > {..} record.json > ...
{
  "Comment": "CREATE record ",
  "Changes": [{
  "Action": "UPSERT",
              "ResourceRecordSet": {
                          "Name": "gueritagabs.cetystijuana.com",
                          "Type": "CNAME",
                          "TTL": 3,
                      "ResourceRecords": [{ "Value":  "http://d2c4shdgmnyt9.cloudfront.net/"}]
  }}]
  }
```

Here we need to use the **UPSERT** action in order to insert and update the already existing file, and the **ResourceRecords** is going to be changed to the CloudFront endpoint instead of the S3 bucket directly. My domain name is: *http://D2c4shdgmnyt9.cloudfront.net,* this will be the only change needed at the moment in this file.

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

this command is used to upload the correct record.json file, if we don't remember our host zone id, we can extract it with the command:

```
aws route53 list-hosted-zones
```

In our case is `Z03346142C3RKH191036Y`, and finally, we just need to add the file path where our record.json is located locally on our machine.

**3. Explain what it means to minify website resources (html, js, css) and the advantages and disadvantages of this process. 10 points.**

Minify like the word sound means to minimize the amount of code in a web page and its script files, this is made to reduce the *bandwidth* (the maximum amount of data transmitted over an internet connection in a given amount of time) and *load times* (how long it takes for a website, or web page, to fully load and appear on screen). One of the main characteristics of **Minify** is that it removes all the comments and extra spaces.

**Advantages:**

- There exist several documentation and optimization processes to minify your projects (for HTML, CSS, JS).
- It improves the loading time (it makes the loading time faster when someone reaches the webpage).
- The users won't download unnecessary data by having a smaller file size.
- It will make your page more user likable and can give a higher SERP ranking (search engine results pages, make your page appear on the top when a search is made)

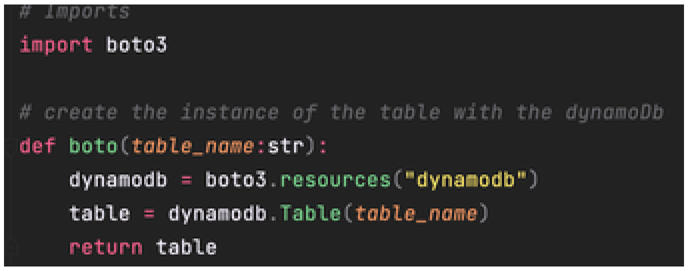
**Disadvantages:**

- One disadvantage of this process is that doing the process manually is not a good practice.
- If you use a tool to make a code  minify, you will need to keep updating the minify version updated since it will continuously be changing.
- It's hard to read, if a developer wants to read the minify version it can be quite a challenge without any line breaks or comments.

**4. Write a python script (and explain each step with great technical detail) to:**

**a) Create or update a record in the Students DynamoDB table based on the id.**

First, we need to import the **boto3** so we can create, configure, and manage AWS services with Python for DynamoDB. Now that boto3 was imported we need to create a connection by initializing dynamoDb with boto3.resource and specifying what table we need, which is "Students".



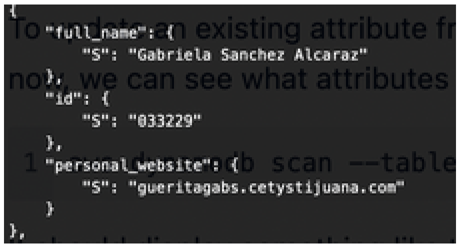
```
# imports
import boto3

# create the instance of the table with the dynamoDb
def boto(table_name:str):
    dynamodb = boto3.resources("dynamodb")
    table = dynamodb.Table(table_name)
    return table
```

For that we use **boto3.resources("dynamodb")**, (it tells the type of service it's going to interact with: Amazon DynamoDB). then we use **dynamo db.Table** that instantiates a table (doesn't creates it), and for our work it would be the "Students" table.

To update an existing attribute from a table or even to add a new attribute we can use .**update_item** now, we can see what attributes we have with the command:

```
aws dynamodb scan --table-name Students
```

It should display something like this:



```
{
    "full_name": {
        "S": "Gabriela Sanchez Alcaraz"
    },
    "id": {
        "S": "033229"
    },
    "personal_website": {
        "S": "gueritagabs.cetystijuana.com"
    }
},
```

Now we can try to update the *id*

```
def createUpdate(tableName, item):
  table.put_item(Item=item)

#example data:
tableName: "Students"
item = {
"id"; '012345'
"Full_name": "Gabriela Sanchez Alcaraz"
"Personal_website": "gueritagabs.cetystijuana.com "
}

studentTable = botoInit(tableName)
createUpdate(studentTable, item)
```

Here we use the **put_item** method that creates a new item or replaces an old item with a new item. If an item that has the same primary key as the new item already exists in the specified table, the new item completely replaces the existing item. (Boto3 documentation), here the only change was the **id** it will be replaced with the new specified parameter.

**b) Delete a record in the Students DynamoDB table based on the id.**

To delete an item Boto3 also has a specific method to do that. We can use **delete_item()** it deletes a single item in a table by primary key.

```
def deleteRecord(table, key):
  table.delete_item(Key=key)

# example data:
tableName: “Students”
key = {“id”: “033229”}

studentTable = botoInit(tableName)
deleteRecord(studentTable, key)
```

This would delete my id number if it matches the same key argument inside the Students table.

**c) Find a record in the Students DynamoDB table by id.d. 20 points**

To find a specific ID inside the Students table, we can use the **get_item()** method, it returns a set of attributes for the item with the given primary key.

```
def findRecord(table, key):
  foundRecord = table.get_item(Key=key)
  print(foundRecord)
  Return foundRecord

#example data
tableName: “Students”
key = {“id”: “033229”}

studentTable = botoInit(tableName)
findRecord(studentTable, key)
```

**5. Explain the difference between a Growth Mindset and a Fixed mindset according to Carol Dweck. 10 points**

| Growth Mindset | Fixed Mindset |
|---|---|
| Believes that intelligence, abilities and talents can be learnable through time and practice. | Believes that intelligence, abilities and talents can not be improved in someones way of being, you are born with them. |

Dweck states that when someone with a **fixed** mindset is set to an extreme situation where they feel intimidated by “knowing” they don’t have the right abilities to solve it, will feel stressed and anxious about the situation, while in contrast, someone with a **growth** mindset will try to either learn from others or put in practice some past knowledge to create a new one.

He points out some reasons why we should be open-minded like the growth mindset such as:

- **You will be able to move to different work fields**: If you need/want a new job you need to gather new skills. And for our career, I believe this is something significant to do, why? because we have a lot of options (cybersecurity, cloud computing, data analysis…) and we always need to be updated of how new tools work.
- **You’ll always be updated:** Being a growth thinker makes you want to makes you keep up with the changes around you and will make you a better worker, your work will always be evolving and you’ll be learning from previous mistakes.
- **It keeps you humble:** By being this open-minded you are willing to accept other may know more than you do, and that means it’s okay to ask for help, and keep going to school/taking courses. Fixed mindset will on the contrary be jealous of others' knowledge and takes feedback as a personal attack.

From what has been researched, this mindset can come from someones childhood, when you are a kid you naturally lean toward the growth mindset since you are curious about your environment and explore and learn through all of your senses, while kids with a fixed mindset will be more likely to interpret failure as evidence of their lack of intelligence or capability, this can be caused due to strict parents or teachers who cause great reprimands.