# Assignment 1

ICE191 Software Architecture / Cloud Computing

T032218 - Elian Javier Cruz Esquivel

**Nota: Las preguntas 4, 5 y 6 las hice en equipo junto a Marco Vélez**

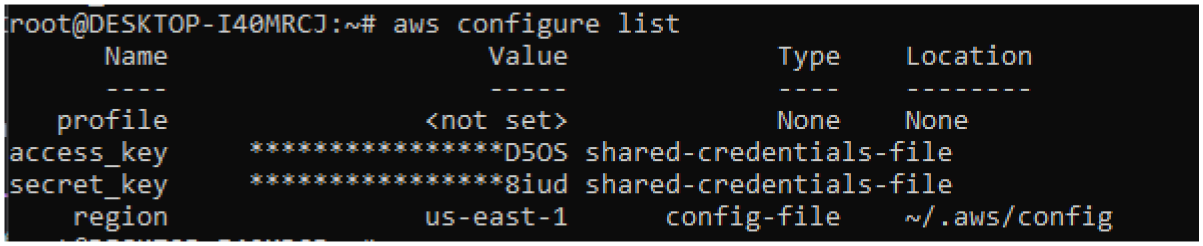
1. Read the following article http://martinfowler.com/articles/microservices.html and explain what "Microservices Architecture" means. 20 points.

   Microservices architecture makes reference to the approach given to the type of systems built upon individual components or modules of code that behave in an independent manner to distribute the logic behind each independent task or logic. Each one of these independent so-called services allow the developers to think of them as a whole,  creating a piece of software that has been designed as a product on its own, taking into account the infrastructure and management that will be required to build it.

   This type of architecture allows that each of these services behave as an goal that provides its teams to properly work as a cohesive entity with cross-functional behavior that provide expertise and knowledge to build the service or feature as better as they can without having to take into account the limitations of its organization or the norms that will be required to integrate this new feature into a monolithic (and sometimes legacy) system.

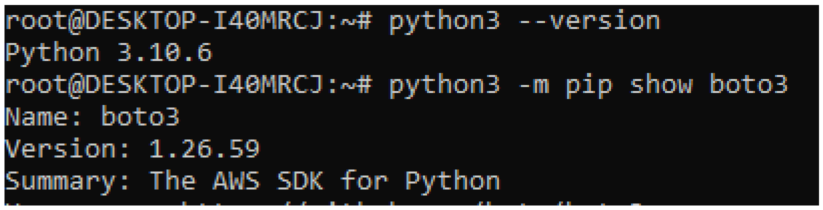
   Another key feature of microservices architecture is the fact that is it decentralized in both teams and infrastructure-wise, therefore, the decisions are not scaled along a chain of command, but are owned by the developers, which take the best decisions to ensure the reliability and success of the system they will be required to build and maintain. This decentralization provides diversity to the environment of the organization by guaranteeing that the best tools and languages are considered and used for each specific problem, making easier the gradual development and evolution of each one of these services, changing them when it's necessary (whether by refactoring or switching to a new set of tools) limiting the amount of overhead needed to satisfy a tool-set required for an organization with another type of architectural organization.

2. Install aws cli in your Mac or Linux computer and setup the config and credentials files. 20 points.



```
root@DESKTOP-I40MRCJ:~# aws configure list
      Name                    Value             Type    Location
      ----                    -----             ----    --------
   profile                <not set>             None    None
access_key     ****************D5OS shared-credentials-file
secret_key     ****************8iud shared-credentials-file
    region                us-east-1      config-file    ~/.aws/config
```

3. Install python and boto3 in your Mac or Linux computer. 10 points

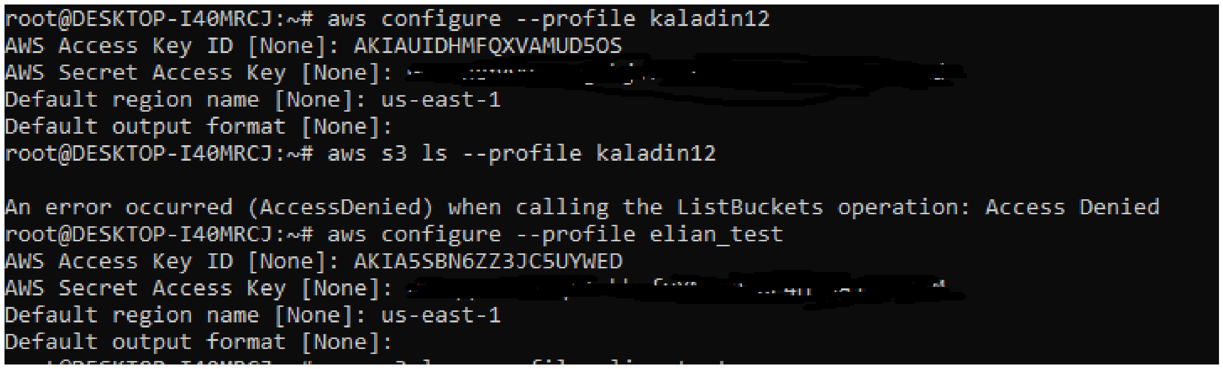


```
root@DESKTOP-I40MRCJ:~# python3 --version
Python 3.10.6
root@DESKTOP-I40MRCJ:~# python3 -m pip show boto3
Name: boto3
Version: 1.26.59
Summary: The AWS SDK for Python
```

4. Explain how you can have more than one access key in your computer and how to use each one. Provide code or configuration examples. 10 points.

En primera instancia, se generan dos o más pares de access keys (Access y Secret) para el usuario IAM. Posteriormente, con cada par de llaves se establece una configuración desde la terminal al llamar al comando *aws configure* utilizando la bandera de *--profile* para establecer un "alias" que identifique a cada configuración (access key).

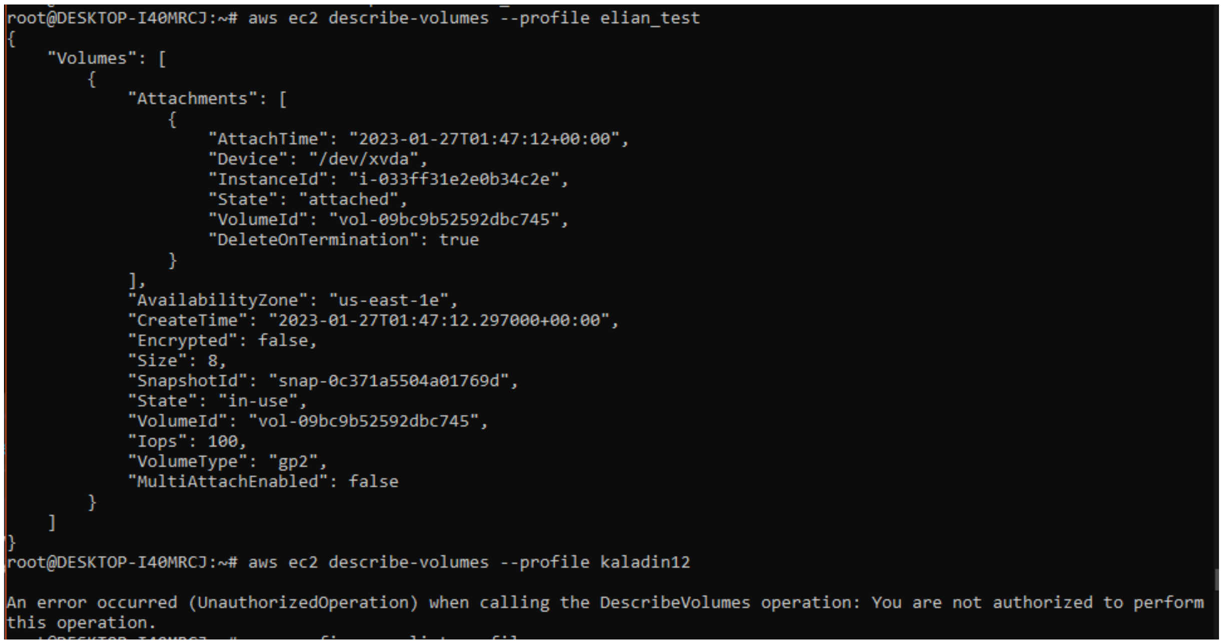
Se introduce el access key, secret access key y la región predeterminada para cada uno de los perfiles. En este caso se generaron dos perfiles denominados *kaladin12* y *elian_test* como nombre de perfil



```
root@DESKTOP-I40MRCJ:~# aws configure --profile kaladin12
AWS Access Key ID [None]: AKIAUIDHMFQXVAMUD5OS
AWS Secret Access Key [None]: [redacted]
Default region name [None]: us-east-1
Default output format [None]:
root@DESKTOP-I40MRCJ:~# aws s3 ls --profile kaladin12

An error occurred (AccessDenied) when calling the ListBuckets operation: Access Denied
root@DESKTOP-I40MRCJ:~# aws configure --profile elian_test
AWS Access Key ID [None]: AKIA5SBN6ZZ3JC5UYWED
AWS Secret Access Key [None]: [redacted]
Default region name [None]: us-east-1
Default output format [None]:
```

Posteriormente, para demostrar el uso e independencia de cada perfil se hace la llamada a determinado comando, en este caso *aws ec2 describe-volumes* definiendo el nombre del perfil que hace la llamada en cada caso. Al hacer la llamada con el perfil *--profile elian_test* es posible observar que se obtiene como resultado un volumen asociado a dicha cuenta, sin embargo, al utilizar *--profile kaladin12,* que corresponde a la cuenta proporcionada por el profesor, es posible observar que no hay permisos suficientes al utiliza las credenciales asociadas a dicho perfil.



```
root@DESKTOP-I40MRCJ:~# aws ec2 describe-volumes --profile elian_test
{
    "Volumes": [
        {
            "Attachments": [
                {
                    "AttachTime": "2023-01-27T01:47:12+00:00",
                    "Device": "/dev/xvda",
                    "InstanceId": "i-033ff31e2e0b34c2e",
                    "State": "attached",
                    "VolumeId": "vol-09bc9b52592dbc745",
                    "DeleteOnTermination": true
                }
            ],
            "AvailabilityZone": "us-east-1e",
            "CreateTime": "2023-01-27T01:47:12.297000+00:00",
            "Encrypted": false,
            "Size": 8,
            "SnapshotId": "snap-0c371a5504a01769d",
            "State": "in-use",
            "VolumeId": "vol-09bc9b52592dbc745",
            "Iops": 100,
            "VolumeType": "gp2",
            "MultiAttachEnabled": false
        }
    ]
}
root@DESKTOP-I40MRCJ:~# aws ec2 describe-volumes --profile kaladin12

An error occurred (UnauthorizedOperation) when calling the DescribeVolumes operation: You are not authorized to perform
this operation.
```

5. Explain what S3 is, how it works and how you can use it to deploy a static website. 10 points.

Based on the AWS docs [1], S3 stands for Simple Storage Service, which is an object storage service (allows to store objects such as images, json files, etc.) on the Cloud created to store and retrieve any volume of data from any location.

It works by storing data as objects (a file and any metadata that describes it) within structures called buckets (a bucket is a container for objects).

When an object like a picture is uploaded to a bucket, such image object will reside inside it and will be available on the region in which the bucket does, being accessible by invoking the bucket URL with the specific path in which the object is located inside the bucket (e.g. https://bucket.s3.us-east-1.amazonaws.com/files/cetys/project/cuttlefish.png).

Each object is identified within a bucket by a key and a version ID (if versioning is on). The key is basically the unique identifier of an object inside a bucket, therefore, the combination of a bucket, object key and a version ID (if enabled) uniquely identifies each object.

S3 provides data consistency by ensuring read-after-write consistency (requester views the changes after the changes have been made) after PUT and DELETE requests, and has a behavior called eventual consistency in which changes might not appear at the beginning but will appear at some time in the future.

In order to deploy a static website, based on [2], it is needed to create a bucket (either by CLI or in the AWS Console), specifying the bucket name, the region it will reside and accept the remaining default settings. After this, properties should be changed to set the bucket to be used as Static website hosting (set Use this bucket to host a website as Enable), select an index document (index.html or the name of the index document that will be uploaded to the bucket in the future).

Then, permissions for the public access should be changed (set Permissions -> Block public access -> Edit -> clear Block all public access), and grant public access by editing the bucket policy . Finally, upload the index file to the bucket (ensuring it has the same name as specified in the configuration for website hosting).

Finally, to test the website, go to Bucket -> Properties -? Static website hosting and use the bucket website endpoint, that should show the index document if entered on a browser.

6. Explain what CloudFront is, how it works and how you can use it to deploy a static website. 10 points.

Based on the AWS documentation [3], CloudFront is a web service that allows you to speed the distribution of both static and dynamic web content (html, css, images, js files) to users through the use of edge locations, basically working as a CDN, allowing each user to access the content by retrieving it from the nearest edge location to them , lowering the latency and increasing reliability and availability of the content due to replication (multiple copies on different locations)

When a user accesses a website stored in CloudFront, it sends a request for an object like an HTML page of an image, CloudFront DNS routes the request to what are called CloudFront POP (Points of Presence) or edge locations, where it decides which of these can best serve the request (based on latency or availability), then CloudFront check the object to see if it’s cached, if not, it requests the content to the origin server (e.g. a S3 bucket), after receiving it, is added to the cache of the POP and forwards it to the original requester.

In order to deploy a static website, a few steps should be followed:

- Load the static website to an AWS S3 bucket and enabling them to be of public access
- Then, create a new CloudFront Distribution with a web delivery method (used for static content), then select the bucket where the content was upload on the first step
    - Add another configuration like an alternate domain name of an SSL certificate

- Then, by using the CloudFront domain name assigned to the distribution, in a browser access the domain name followed by the static page path: *http://domain name/object name*

7. Explain what an API is. 10 points.

According to IBM [4], it stands for Application Programming Interface, which is basically a set of rules that allow a third party user access or make use of the services from another application without actually knowing or having to understand the internal logic and behavior of its implementation. Therefore, an API only exposes an interface that the rest of the world can use, just like in a GUI, where the interface is graphic and abstracts the inner logic, an API exposes endpoints that are the means by which someone can interact with the application through HTTP calls.

The endpoints are defined by an URI which specifies the exact service that is requested from the third party, and includes a request verb (GET, POST, PUT), headers (permissions, certificates, format) and sometimes a request body. This info is then processed by the application attached to the API, and responds to the third party app by acknowledging the result of the request, either it was successful or not or if there was an error in the process, while the whole inner working of the application is basically transparent to the user.

8. Explain what a Restful API is. 10 points.

Based on [5], it is a type of API that follow 6 principle requirement or architectural constraints:

- Uniform interface: Makes sure that all requests for the same resource or endpoint look the same independent of where comes or who makes the request, therefore, the logic for that specific service belongs to only one URI.
- Client-server decoupling: Both should be completely independent, hence the only information that the client needs from the server can and should be only accessed through the URI, this constraint allows to isolate the behavior of both the client and server.
- Statelessness: This makes reference to the fact that each request includes all the information necessary for a processing task to take place on the server side, without the need of the server to store this client data.
- Cacheability: Refers to the fact that a client request for the same resource can be accessed by the client without needing to call the server by storing it in itself through a cache mechanism. In this sense, when the same request is needed over and over again, there won’t be any overhead of having to repeat the same process on the server side for the same information.

- Layered system architecture: Allows to have intermediate servers between the client and the server that provides the API service without affecting the functionality of the request or response, therefore, the client does not know the amount of layers its request has to pass through to get to and from the final server, allowing to place tools like caching or load balancers in the way)
- (Optional) code on demand: Refers to the fact that the server can add functionality to the client by sending code that can be executed by the client, just like a browser, which receives HTML from the server, but also JavaScript code to enhance and add logic that runs in the client side.

9. Explain what you are going to do differently this semester based on last semester experience. 10 points.

Based on the experience from the last semester, I think I need to have a steadier and stricter balance between school and work, especially when it comes to giving that little extra required to actually understand and learn something that is taught in school. Last semester I think that I gave too much of my spare time to work and set aside projects and side hustles that I like to do when I find something interesting in the semester's classes, and I understand that this behavior affected my performance and the quality of learning that I know I acquired from the last semester.

References

[1] "What is Amazon S3? - amazon simple storage service," *What is Amazon S3?* [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html.

[2] "Tutorial: Configuring A static website on Amazon S3," *AWS*. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/HostingWebsiteOnS3Setup.html.

[3] M. Lahtela and P. (P. Kaplan, "Entregar contenido con mayor rapidez," *Amazon*, 1966. [Online]. Available: https://aws.amazon.com/es/getting-started/hands-on/deliver-content-faster/?pg=ln&sec=hs.

[4] "What is an application programming interface (API)," *IBM*. [Online]. Available: https://www.ibm.com/topics/api.

[5] "What is a rest api?," *IBM*. [Online]. Available: https://www.ibm.com/topics/rest-apis