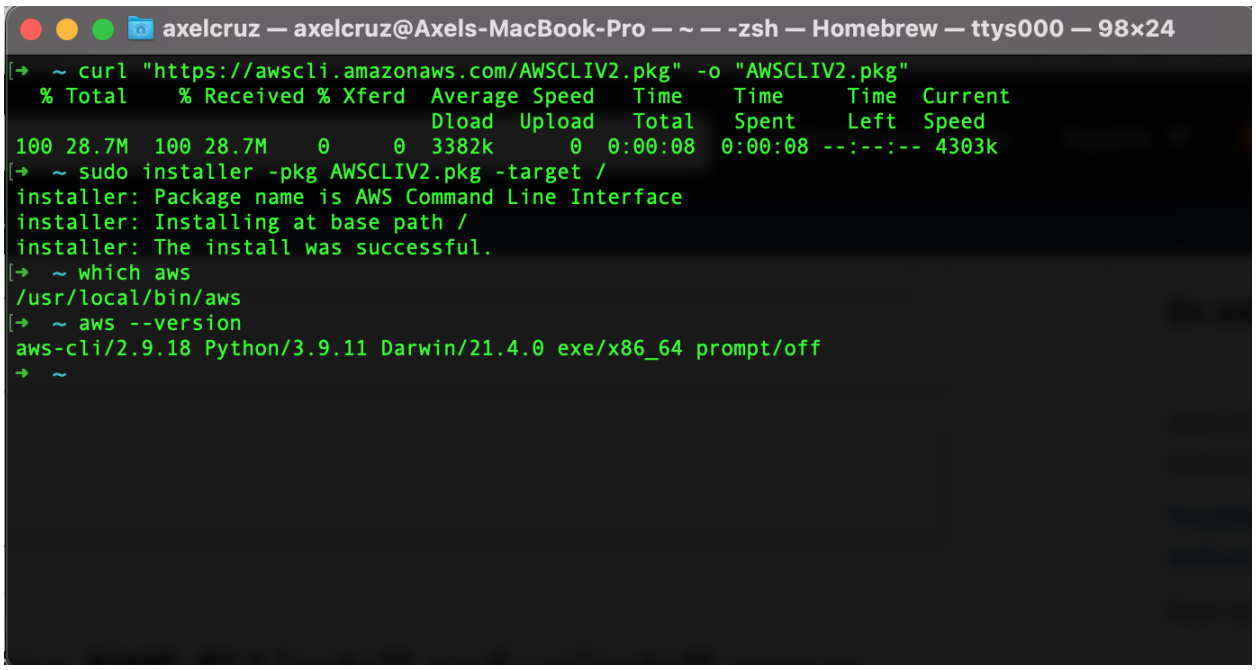
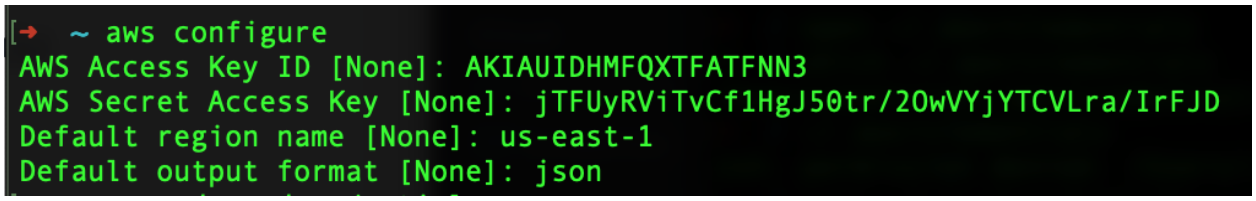
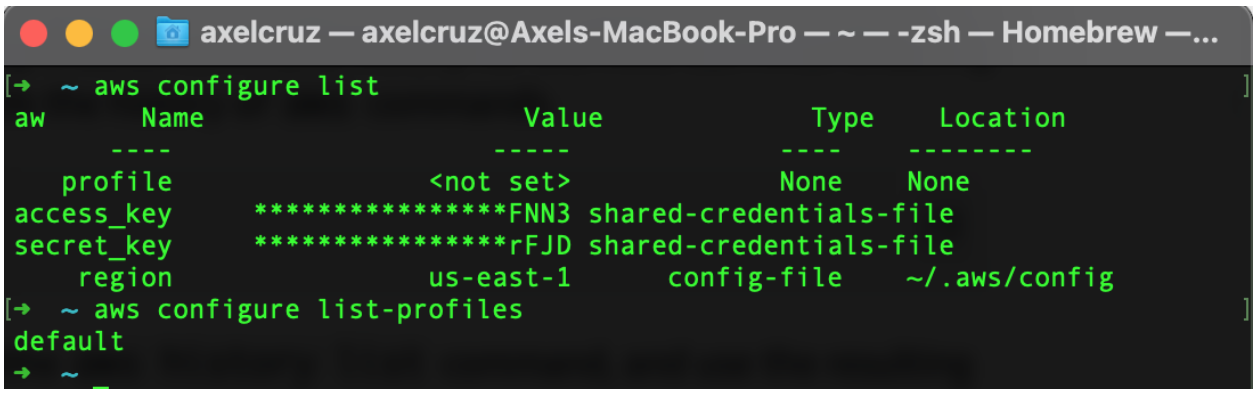
Jorge Axel Cruz Jiménez - T031973

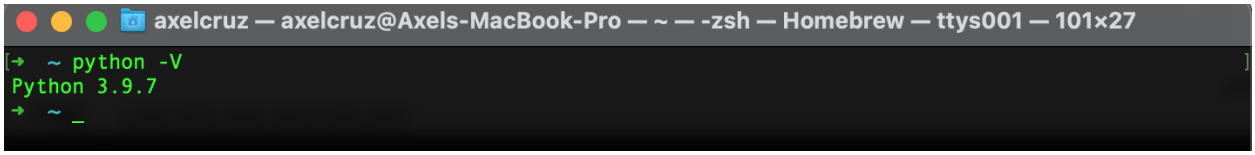
1. Microservices Architecture is a software design approach that allows an application to be developed as a set of small services, so it is broken down into small pieces that work independently from one another that can be maintained, tested, or even scaled, by having small risk of being affected by themselves. Each one of these services run in its own process and communicate with other services independently using APIs (Application Program Interface), by using HTTP request-response protocols. This also allows services to be written in different languages and use different databases depending on the business needs and logic.
2. **Installation of AWS CLI**

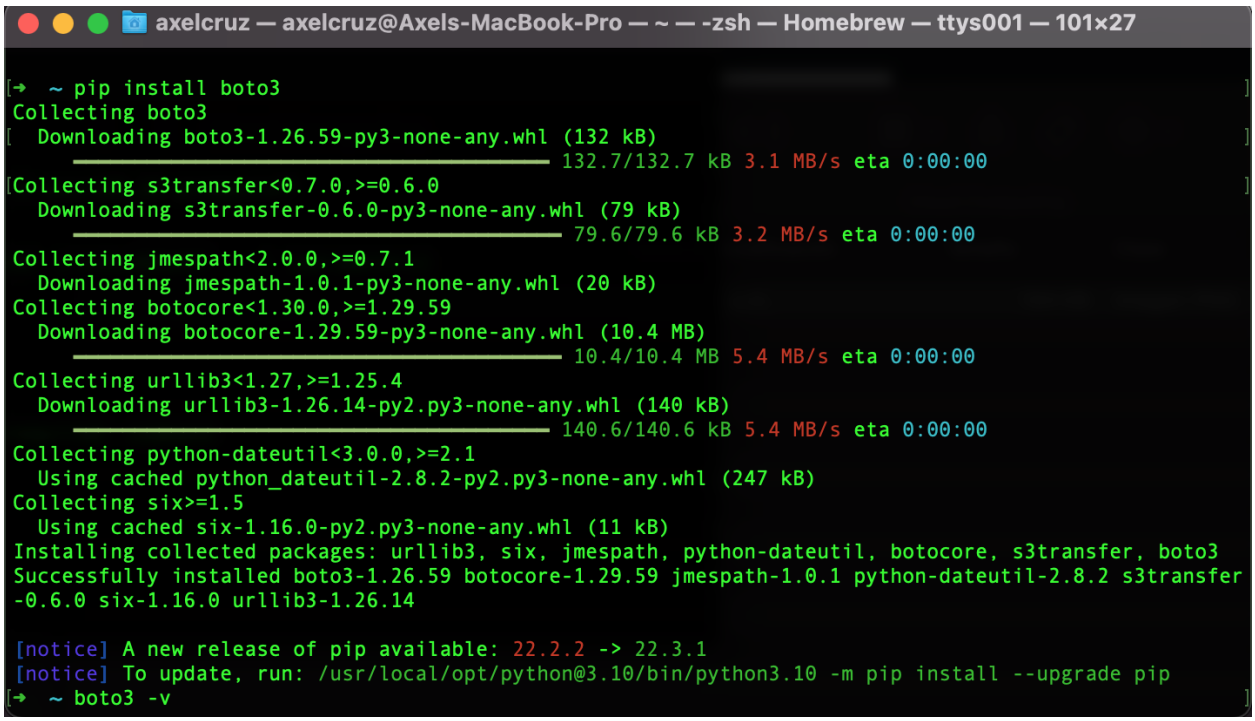


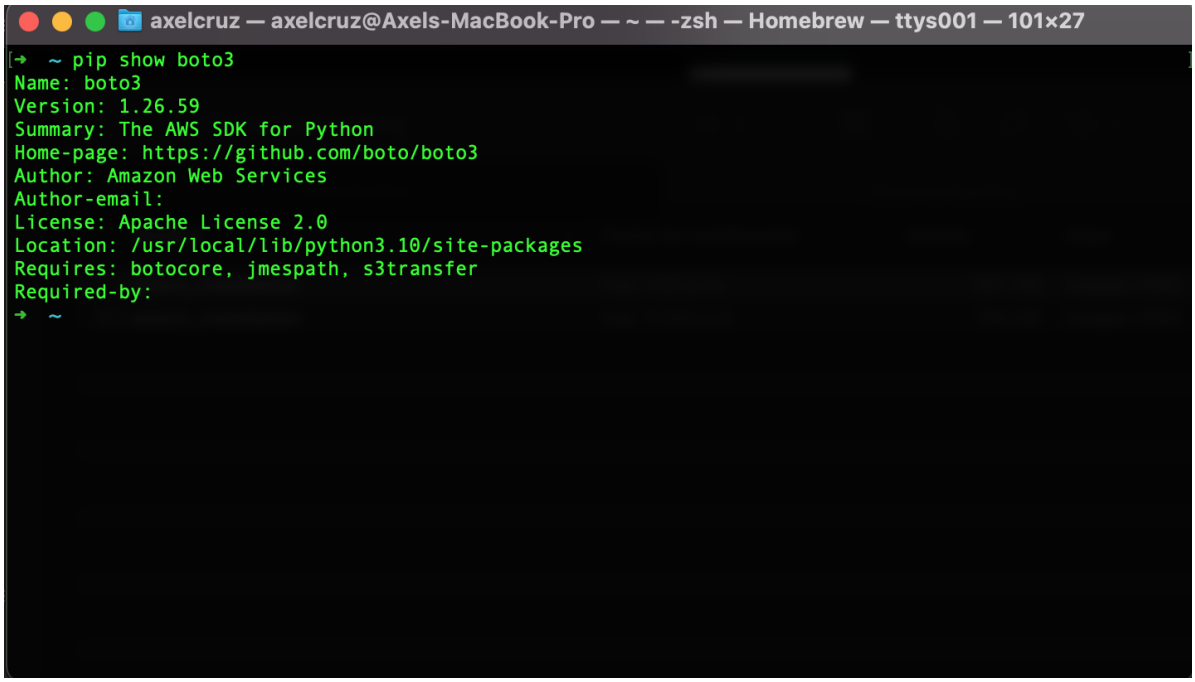



## 3. Python and Boto3



4. **Multiple access keys**

Using named profiles in AWS. They are a collection of settings and credentials that can be applied to a AWS CLI command and multiple named profiles can be stored in the *config* and *credentials* files and each of them have different credentials.

We can specify one default profile when no other profile is explicitly referenced and other profiles have names that can be specified as parameters for individual commands.

To configure additional profiles use *aws configure* with --profile command, or manually add entries in the *config* and *credentials* files.

Example: *credentials* file with 2 profiles.

Access in prompt to file using **~/.aws/credentials**

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

[user1]
aws_access_key_id=AKIAI44QH8DHBEXAMPLE
aws_secret_access_key=je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
```

Each profile can specify different credentials and different AWS Regions and output formats. Include “profile” prefix when naming a profile in *profile* file.

Access in prompt to file using **~/.aws/config**

```
[default]
region=us-west-2
output=json

[profile user1]
region=us-east-1
output=text
```

To use a named profile, add --profile *profile-name* option to the command.

Example: List all Amazon EC2 instances using the credentials and settings defined by user1.

```
$ aws ec2 describe-instances --profile user1
```

Another example could be listing all S3 instances by user1, or another user, using a similar command.

To use a named profile for multiple commands, avoid specifying the profile in every command by setting the AWS_PROFILE environment variable at the command line.

```
$ export AWS_PROFILE=user1
```

5. **Amazon S3**

Amazon Simple Storage Service, is an object storage service that offers scalability, data availability, security and performance. Customers can store and protect any amount of data for virtually any use case such as data lakes, cloud-native applications and mobile apps. Additionally, S3 has various features to organize and manage data in ways that supports specific use cases, enable cost efficiencies, enforce security and meet compliance requirements. Data is stored as objects within resources called "**buckets**", and a single object can be up to **5 terabytes in size**. Features include capabilities to append metadata tags to objects, move and store data across the S3 Storage Classes, configure and enforce data access controls, secure data against unauthorized users, run big data analytics, monitor data at the object and bucket levels, and view storage usage and activity trends across your organization. Objects can be accessed through S3 Access Points or directly through the bucket hostname.

Features:

- Storage Management and security
    - Storage Monitoring
- Storage analytics and insights
    - S3 Storage Lens
    - S3 Storage Class Analysis
- Storage classes
- Access Management and security
    - Access Management
        - IAM
        - Access Control Lists (ACL)
        - Bucket policies
        - S3 Access points
        - Query String Authentication
        - Audit Logs
    - Security
        - Amazon Virtual Private Cloud (Amazon VPC)
        - S3 Inventory
        - S3 Block Public Access
        - S3 Object Ownership
        - IAM Access Analyzer for S3
        - AWS Private Link
        - Amazon Macie
- Data Processing
    - S3 Object Lambda
- Query in place
    - Amazon Athena
    - S3 Select
    - Amazon Redshift Spectrum
- Data Transfer
    - AWS Storage Gateway

        - AWS DataSync
        - AWS Transfer Family
        - Amazon S3 Transfer Acceleration
        - AWS Snow Family
            - AWS Snowball
            - AWS Snowmobile
        - AWS Partner Network (APN)
- Performance
- Consistency

To deploy a static website: create a S3 bucket and configure it as a static hosting location. Then upload the files of the website ( HTML, CSS, and JavaScript) to the bucket and set the permissions for those files so they can be publicly accessible. Then set an endpoint of the bucket (URL) so the website can be accessed.

6. **Amazon CloudFront**

   Amazon CloudFront is a content delivery network (CDN) service built for high performance, security, and developer convenience. It is a distributed servers system to deliver web content to the users based on their geographic location.

   It caches the content of a website that is contained at multiple locations and delivers the content to users that are closed to the server improving the performance and speed of the service.

   CloudFront can be used to distribute content like text, images, and videos. using another AWS service such as Amazon S3 as well and integrates other services for security and protection like AWS Shield or AWS WAF for web applications firewall.

   There are basic forms to deliver content with CloudFront: Create a simple distribution or a secure static website.

   **CloudFront Distribution -** features:

   - Stores the original versions of your objects in an Amazon S3 bucket
   - Makes objects accessible to everyone
   - Uses the CloudFront domain name in URLs for the objects (example: https://d111111abcdef8.cloudfront.net/index.html
   - Keeps objects in CloudFront edge locations for the default duration of 24 hours (minimum is 0 seconds)

   Steps:

1. Upload content to Amazon S3 and grant object permission

   S3 is a container for files or folders. CloudFront can distribute almost any type using a S3 bucket as the source.

   **To upload content to Amazon S3 and grant read permission to everyone**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3
2. Choose Create bucket
3. Enter a bucket name. **The name must conform to DNS naming requirements.**
4. Choose AWS Region. Recommended the closest to optimize latency and minimize costs, or choose another to address regulatory requirements.
5. Clear the checkbox for *Block all public access* in the Block Public Access settings for buckets.
6. Click create bucket.
7. In buckets section, choose the desired bucket and upload.
8. In Access Control List section, select for *Read* next to *Everyone (public access).*
9. Accept the changes and upload.

   After the upload is completed, will get the URL. For example:

```
https://<bucket name>.s3-<AWS Region>.amazonaws.com/<object name>
```

Replace the appropriate values based on the bucket and content.

For region us-east-1, omit the <AWS Region> portion of the URL. For example:

```
https://<bucket name>.s3.amazonaws.com/<object name>
```

Check if the content is publicly accessible using Amazon S3, but that is not the URL that will be used to access the content with CloudFront.

**2.** Create a CloudFront distribution.

1. Open CloudFront console https://console.aws.amazon.com/cloudfront/v3/home
2. Create distribution
3. For Origin domain, choose Amazon S3 bucket
4. For Default cache behavior, use default values
5. For settings, use default values
6. Create distribution
   Save the domain name that CloudFront assigns to your distribution, which appears in the list of distributions. It looks like `d111111abcdef8.cloudfront.net`

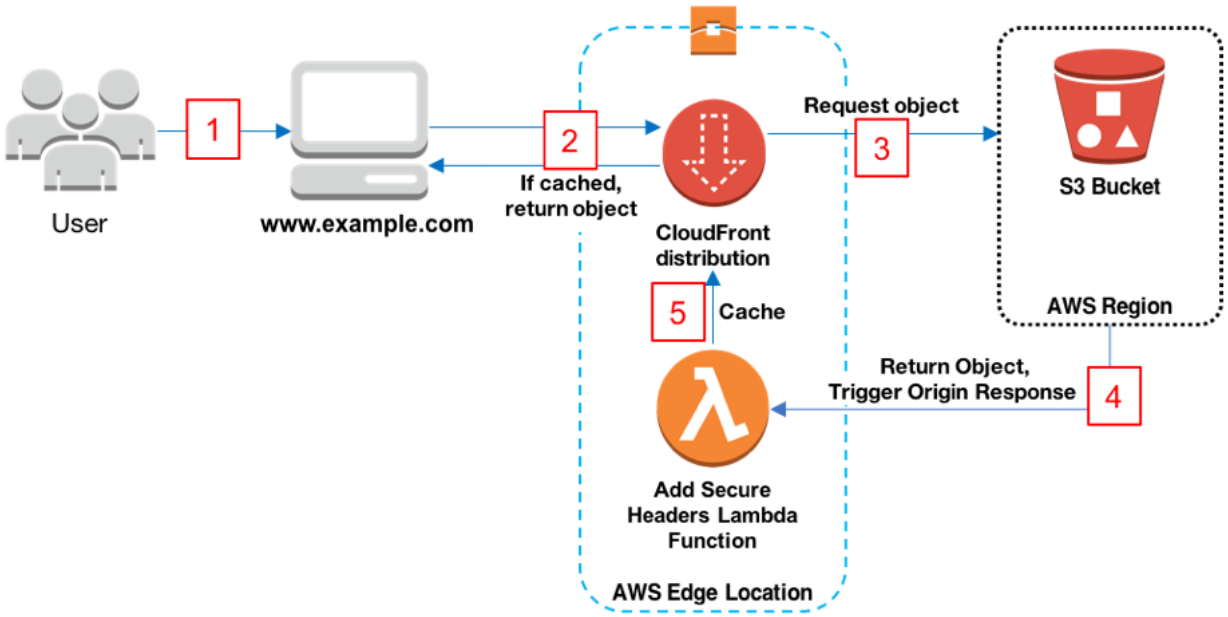
**3.** To access content through CloudFront, combine CloudFront distribution domain name with the path to access the content. For example `https://d111111abcdef8.cloudfront.net/index.html` being the CloudFront path and the content path.
When upload new content, simply follow the same logic, using the CloudFront path and the content of the S3 bucket path. For example: `https://d111111abcdef8.cloudfront.net/new-page.html`

**Secure Static Website**

Creating a secure static website using this method, provides with some benefits:

- The website uses the durable storage of the Amazon S3. It creates a bucket to host the static website just by uploading the files to the S3 bucket.
- The website is sped up by the amazon CloudFront content delivery network. Creates a distribution to serve the website to viewers with low latency. The distribution is configured with an origin access identity to make sure that the website is accessible only through CloudFront, not directly from S3.
- The website is secured by HTTPS and additional security headers. It creates an SSL/TLS certificate in AWS Certificate Manager (ACM) and attaches it to the CloudFront distribution. The certificate enables the distribution to server the domain's website securely with HTTPS.
- The website is configured amd deployed with AWS CloudFormation. Uses an AWS CloudFormation template to set up all the components, so the developer can focus more on the website's content and less configuring components.

Overview




- The viewer requests the website
- If the requested object is cached, CloudFront return the object from its cache
- If the object is not in CloudFront's cache, it requests the object from the origin (an S3 bucket)
- S3 returns the object to CloudFront which triggers the Lambda@Edge origin response event
- The object, including the security headers added by the Lambda@Edge function, is added to CloudFront's cache
- The object is returned to the viewer

When deploying the website using this method, there are different options:

- Use the AWS CloudFormation console to deploy the solution with default content, then upload the website content to Amazon S3
- Clone the solution to the user's computer to add the website content. Then deploy the solution with the AWS Command Line Interface (AWS CLI)

**Using CloudFormation Console**

1. Open the CloudFormation Console.
2. The wizard opens the console with prepopulated fields that specifies a template.
3. On the Specify stack details page, must enter values for fields:
    a. **SubDomain** - Enter the subdomain for the website (www.)
    b. **DomainName** - Enter domain name. Must be pointed to a Route 53 hosted zone.
4. Configure stack options
5. Create a IAM role to allow access to the stack's resources and name them dynamically
6. Create stack.
7. When finished view website by subdomain and domain.

**Cloning the solution locally**

1. Clone or download the solution from https://github.com/aws-samples/amazon-cloudfront-secure-static-site. After download open command prompt or terminal and navigate to the `amazon-cloudfront-secure-static-site` folder.
2. Run the command to install and package the artifacts: `make package-static`
3. Copy website's content into the www. Folder, overwriting the default website content.
4. Run following command to create an Amazon S3 bucket to store the artifacts. Replacing with own bucket name.

```
aws s3 mb s3://example-bucket-for-artifacts --region us-east-1
```

5. Run following AWS CLI commands to package artifacts as a CloudFormation template. Replacing with own name bucket.

```
aws cloudformation package \
    --region us-east-1
    --template-file templates/main.yaml \
    --s3-bucket example-bucket-for-artifacts \
    --output-template-file packaged.template
```

6. Run following command to deploy with AWS CloudFormation, replacing with own values.
*your-CloudFormation-stack-name* - Replace with own AWS CloudFormation stack.
*Example.com* - Replace with own domain name. Pointed to a Route 53 hosted zone.
*www* - Replace with own subdomain name.

```
aws cloudformation deploy \
    --region us-east-1
    --stack-name your-CloudFormation-stack-name \
    --template-file packaged.template \
    --capabilities CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND \
    --parameter-overrides DomainName=example.com SubDomain=www
```

7. Wait for stack creation and go to the website's content using own subdomain name and domain name.

7. **API**

   Application Program Interface, is a method, or set of rules, that is used for building or interacting with software applications. APIs are built to specify how software components in applications should interact with each other and this allows for communication between multiple systems. This allows data to be passed from one application to another, or from one system to another, or even for a system to request a service from another. For example, Social media platforms such as Facebook, Twitter and Instagram have APIs that allow the developers to access certain features of the platforms. For example, a developer can use the Facebook API to retrieve another user's profile information or to post updates on their timeline.

   Another example is that E-commerce platforms such as Shopify, Amazon and Ebay provide APIs to access product information, create and manage orders, and handle payments.

8. **Restful API**

   APIs that uses the REST architectural style and constraints. REST, or Representational State Transfer, is an architectural style for building web services, so the can be more scalable, maintainable, and flexible and are characterized by a client-server architecture, stateless communication and a uniform interface.

   Restful APIs are typically build using HTTP and use the following HTTP methods:

   - GET: retrieve information from a server
   - POST: submit new information to a server
   - PUT: update existing information on a server
   - DELETE: delete information from a servera

   These APIs use a structure for the URLs that are used to access resources known as a resource-oriented architecture, where resources are identified by URIs, and are accessed using HTTP methods.

   Additionally, they use JSON format for sending and receiving data, and another methods such as XML among others.

   Restful APIs are widely used in web and mobile applications, due to simplicity and flexibility, they are also easy to test and support caching easily, which improves performance.

9. This semester I'm going to do more self learning practice topics that I'm new at or just have heard about but never actually dug into them. The main reason is so I can expand my knowledge and try to find the path and field in which I want to develop as a software student. Additionally I will look for activities that can help me to improve my critical thinking.