Managing and Hosting a Web Server using AWS

Abdul A
Marcus H
Garrett S

4-26-2023
Spring 2023
Cloud Computing

Table Of Contents

EXECUTIVE SUMMARY **3**

PROJECT SCHEDULE MANAGEMENT **6**

MILESTONE 1: WEBSITE SETUP (S3, EC2, IAM) **7**

MILESTONE 2: SETTING UP CLOUDWATCH AND BACKUP **11**

MILESTONE 3: THE FINAL OUTCOME **14**

MILESTONE 4: ARCHITECTURE **15**

REFERENCES **17**

EXECUTIVE SUMMARY

In our project we used Amazon Web Services to host a web application which any user could access to gain knowledge and insight on Cybersecurity certifications. This entailed making use of seven AWS services which allowed the website to be hosted and accessed. We focused on core cloud services AWS has to offer, to gain a better fundamental understanding of how exactly the process works in setting up and hosting a web application, rather than relying on a service like Amazon LightSail, which would do a lot of the heavy lifting in the deployment of the web application, leaving us wondering how exactly the process worked. The services we used in the setup and deployment of our web application were IAM, EC2, S3, CloudWatch, Billing, Backup and Cloud Shell. It should be noted that the Billing and Cloud Shell services were not necessarily a key component in the final goal of our project, but served as useful tools, with Billing letting us know our budget expenditure, and Cloud Shell as a CLI access for easy and fast setup for IAM roles and policies, EC2 instance creation, and S3 bucket creation. The initial process for setting the website up started with our core services, IAM, EC2, S3, which was executed with Cloud Shell (CLI). The first step involved creating a bucket with the S3 service. Once the bucket was created, we simply uploaded our HTML zip file into the bucket, which could eventually be retrieved by the EC2 instance. The next step involved creating an IAM role with S3FullAccess policy attached to it, which we could then create an instance profile and attach this role to it. This instance profile serves the purpose of passing a role to our EC2 instance, which we create next. Once the Instance was live, we decided to do the process of retrieving the html files manually, by connecting to the instance using SSH. Once connected to the instance, we ran wget with the link to the S3 bucket, and since the instance has the instance profile with S3FullAccess, the file was successfully retrieved. Next was moving the zip file to the /var/www/html directory on the system, and unzipping. At this point, all to be done was installing and starting httpd, and the website was successfully up and running. After the web application setup, we connected a CloudWatch agent to the instance to monitor logs and network traffic to the website, as well as configuring a monthly backup of the instance using the AWS Backup service, in case of instance failure. Lastly, we set up an AWS billing budget and notification system to monitor our budget expenditure. By implementing all these services, we were able to create a secure and reliable hosting environment for our web application, while simultaneously gaining hands-on experience and knowledge using AWS. This experience lays a nice foundation for our future endeavors involving the Cloud.

Objective

The aim of this project was to deploy and host a web application on Amazon Web Services (AWS) using an EC2 instance. We utilized the S3 service to store the website files, and installed a CloudWatch agent to monitor logs and network traffic. Additionally, we set up an AWS billing budget and notification system to manage costs effectively, and configured AWS Backup to automatically perform backups every month. By implementing these features, we were able to create a secure and reliable hosting environment for the web application, while also ensuring that costs were managed efficiently. This project demonstrated our ability to effectively utilize AWS services to deploy and manage web applications.

Background

In cloud computing class we have learned about the numerous services that AWS has to offer. We wanted to utilize seven of these services, and those being AWS IAM, AWS EC2, AWS S3, AWS Cloud Shell, AWS CloudWatch, AWS Backup, and AWS Billing. The way we used these seven services allowed us to make our own web application and have it run successfully.

Conclusion

Aside from getting hands on learning with Amazon Web Services, this project has allowed us to have a better understanding and work more in depth with the numerous services that they have to offer. The use of AWS allowed us to deploy and host a website all from scratch with just the use of seven AWS services. The main steps it took to complete this project were creating the EC2 instance, the S3 bucket, giving access to it, and installing CloudWatch agent. These steps allowed us to have our web application run efficiently and effectively.

Project Milestones:

1. Website setup (S3, EC2, IAM)
2. Setting up CloudWatch and Backup
3. The Final Outcome
4. Architecture

Materials List:

1. Computer
2. AWS Services
3. Visual Studio

Deliverables:

1. PowerPoint presentation
2. Report

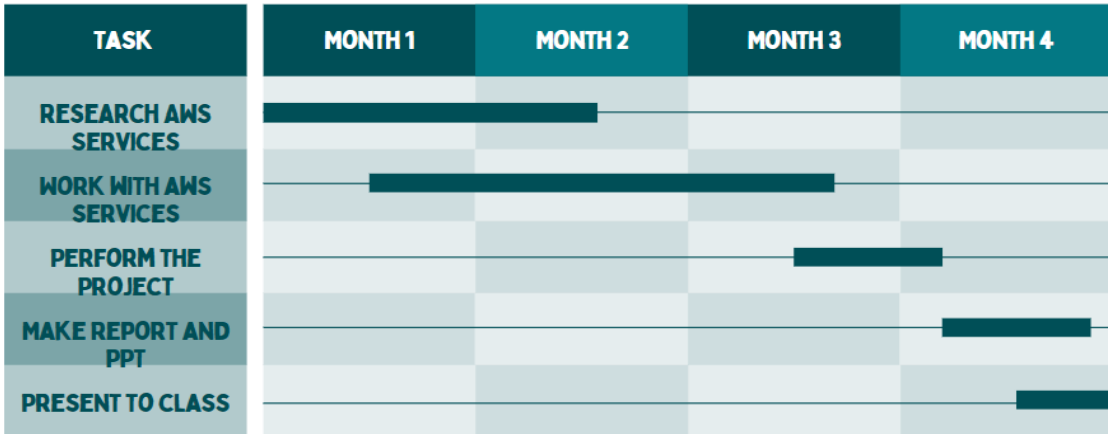
3. Web application

Professional Accomplishments:

1. Experience in setting up and deploying a web application on the AWS infrastructures
2. Knowledge of using AWS services such as IAM, EC2, S3, CloudWatch, AWS Backup, and AWS Billing
3. Learned how to work together as a team and apply all our skills together to reach one common end goal

PROJECT SCHEDULE MANAGEMENT

GANTT CHART



We did not have access to a Trello board because it asked us to pay for its services.

https://github.com/alqarni360/cyber-security-projects/blob/4f701668c3a673fb970ed5733b61dbbd6cdd5e13/UIW%20Final%20Project.pptx.

Milestone 1: Website Setup (S3, EC2, IAM)

Amazon S3 (Simple Storage Service) is a highly scalable and durable object storage service that allows users to store and retrieve data from anywhere on the web. For this reason, we chose Amazon S3 to deposit our web application files, which could be retrieved later by the EC2 instance. The first step in the deployment of the web application is creating an S3 bucket and uploading the compressed zip file, which contains all the necessary files for our html page to run correctly. To create the S3 bucket, you need to choose a globally unique name for your bucket and specify the region in which you want to create it. We did this using Cloud Shell, with the -aws s3api create-bucket command.

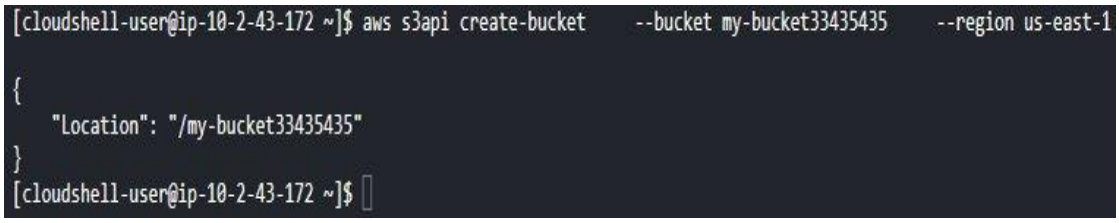


Figure 1.0, creating bucket.

This was a fast and easy S3 bucket creation, and now we can move on to depositing our web application files into the bucket. Before executing the command, we must upload the zip file of our web application into the Cloud Shell directory. Once that is done, we can deposit the files into the bucket using the -aws s3 cp command using the CLI.

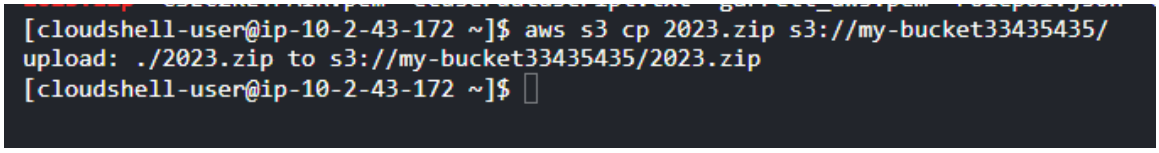


Figure 1.1, copying files to bucket.

Now that our web application zip file is deposited into the bucket, we are a step closer to getting the website up and running. The next steps involve creating a role and attaching an S3FullAccess policy to it, to then create an instance profile and attaching that role to it. The purpose of this is to have permissions properly setup for our EC2 instance, which has not been created yet, to access our S3 bucket when it is created. This step is crucial, and without it our files in the S3 bucket will not be able to be accessed or retrieved by the instance. The role can be created with the command aws iam create-role. An important step is having a trust policy attached to the role, which specifies that the role is meant for EC2 service use. We created trust policy .json file, ec2s3.json, which we uploaded into Cloud Shell to be used for attachment to the role.

```
[cloudshell-user@ip-10-4-14-72 ~]$ aws iam create-role --role-name ec2s3fin --assume-role-policy-document fi
le://ec2s3.json
{
    "Role": {
        "Path": "/",
        "RoleName": "ec2s3fin",
        "RoleId": "AROA2AFW3QCPCZF4F3HEC",
        "Arn": "arn:aws:iam::687578251422:role/ec2s3fin",
        "CreateDate": "2023-04-19T00:24:10+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "ec2.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
[cloudshell-user@ip-10-4-14-72 ~]$ aws iam attach-role-policy --role-name ec2s3fin --policy-arn arn:aws:iam::aws
ll-user@ip-10-4-14-72 ~]$ aws iam attach-role-policy --role-name ec2s3fin --policy-arn arn:aws:iam:
:aws:policy/AmazonS3FullAccess
[cloudshell-user@ip-10-4-14-72 ~]$
```

Figure 1.2, creating role and attaching policy.

Here the role is created and named, with the trust policy attached to it. After, we used the command -aws iam attach-role-policy command to attach the AmazonS3FullAccess to the role.

With this complete, now we must create an instance profile to pass this created role to our EC2 instance, which will then give the instance permissions to retrieve data from the S3 bucket. We cannot simply pass this role directly to our instance, it must be through an instance profile. The creation of the instance profile can be done using the -aws iam create-instance-profile command, and the passing of the role to the instance profile can be achieved with the -aws iam add-role-to-instance-profile command.

```
[cloudshell-user@ip-10-4-14-72 ~]$ aws iam create-instance-profile --instance-profile-name ec23fin-Instance-
Profile
{
    "InstanceProfile": {
        "Path": "/",
        "InstanceProfileName": "ec23fin-Instance-Profile",
        "InstanceProfileId": "AIPA2AFW3QCPK2V4FOKQW",
        "Arn": "arn:aws:iam::687578251422:instance-profile/ec23fin-Instance-Profile",
        "CreateDate": "2023-04-19T00:30:52+00:00",
        "Roles": []
    }
}
[cloudshell-user@ip-10-4-14-72 ~]$ aws iam add-role-to-instance-profile --role-name ec2s3fin --instance-prof
ile-name ec23fin-Instance-Profile
[cloudshell-user@ip-10-4-14-72 ~]$
```

Figure 1.3, creating instance profile and attaching role.

Once this is successfully achieved, we can move on to creating the EC2 instance. This is accomplished with the -aws ec2 run-instances command. All the details encompassing the creation of the instance must be included, such as the instance type, and the count.

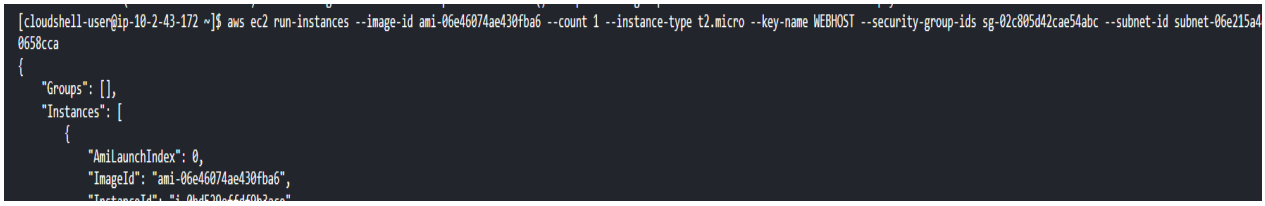


Figure 1.4, creating EC2 instance.

With the instance created successfully, now we must associate the instance profile we created earlier to the instance. This can be done with the -aws ec2 associate-iam-instance-profile command.

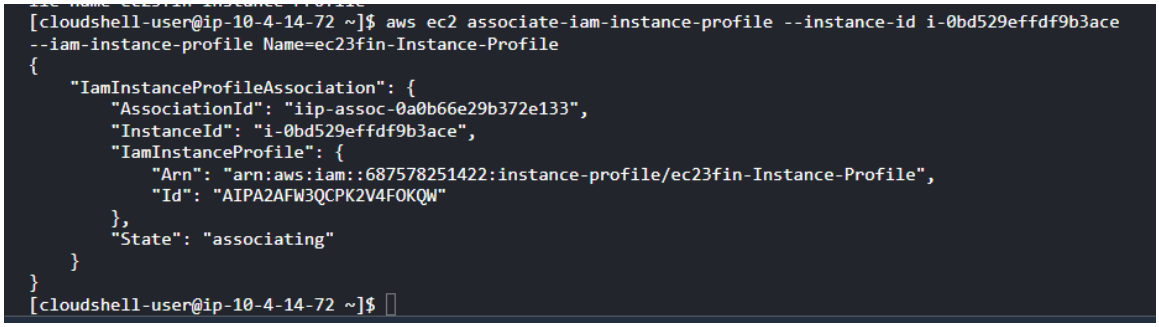


Figure 1.5, associating instance with instance profile.

Now that the instance profile is associated with the instance, our EC2 instance can successfully communicate with the S3 bucket and can retrieve the files needed for the web application. Now all we must do is some cleanup work on the EC2 instance itself. For this we connect using ssh. Once connected to the system, we can run the -wget command to copy the zip file from the bucket.

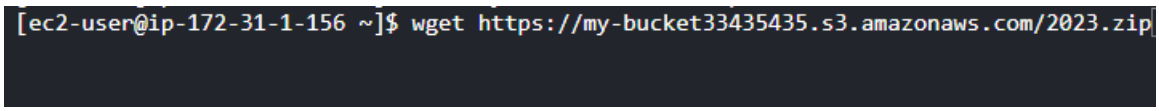


Figure 1.6, retrieving files on instance from bucket.

The bucket zip file is now on our instance, and now it must be unzipped and moved to the /var/www/html directory.

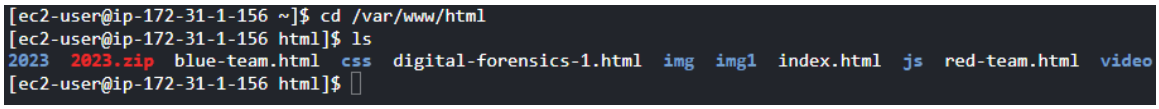


Figure 1.7, moving files to /var/www/html.

With the files in the proper directory, all that must be done now is to install the httpd service and start it. In retrospect, this could have been accomplished by passing a user script to the instance to automatically do this on start-up, but we decided to do this step manually.

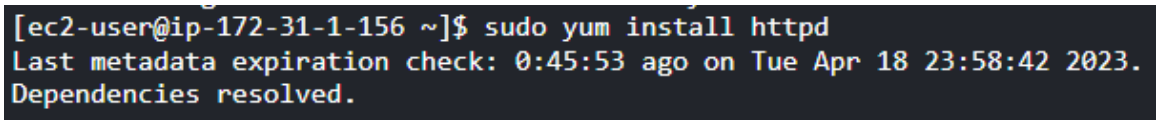


Figure 1.8, installing httpd.

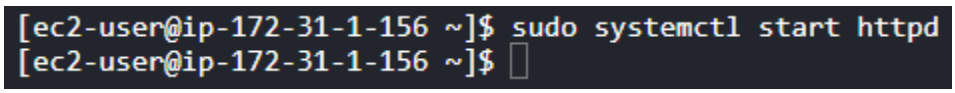


Figure 1.9, starting httpd.

After httpd is started, our web application is now up and running, and can be accessed by connecting to the public ipv4 address of the EC2 instance.

Milestone 2: Setting up CloudWatch and Backup

After completing the setup of the web application, we can configure AWS CloudWatch, this is a monitoring service that allows us to see what traffic comes into and out of our website. With the use of this service, we can collect data in the form of logs. AWS CloudWatch allows us to also store this data so we can look at and analyze it any time we want, the data is stored in chronological order with the most recent data and information appearing first. The next thing we can do with CloudWatch is to be able to analyze the information that is given to us, whether if that is using a specific visual or using a table to help you do that, the choice is yours. The picture below shows the traffic that is coming into and out of our website.

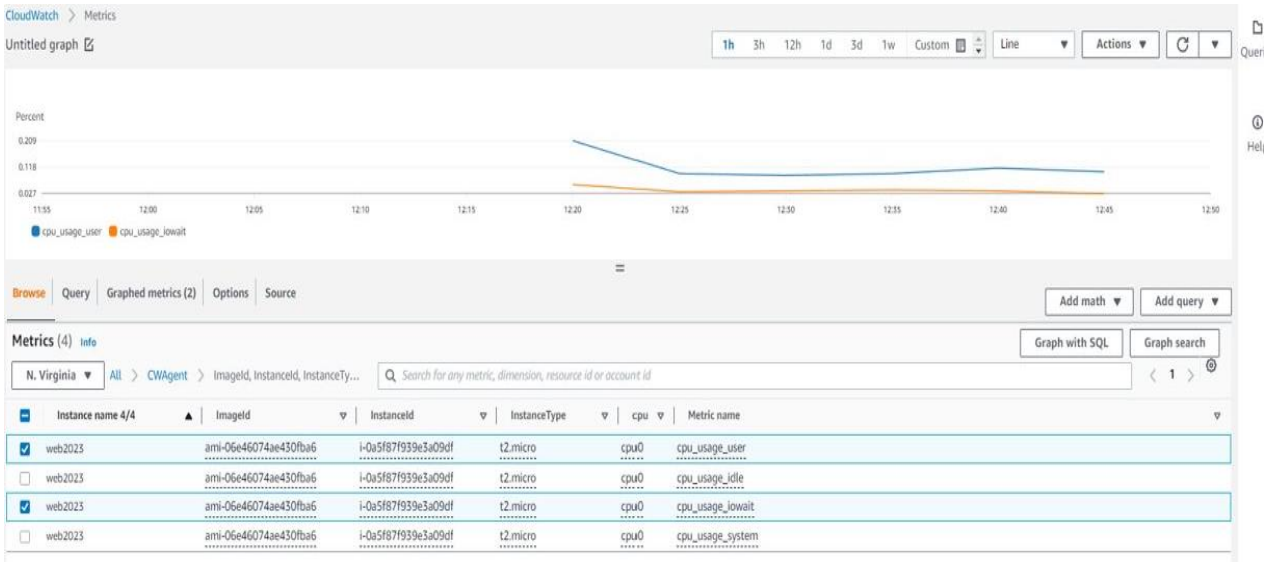


Figure 2.0, traffic to website.

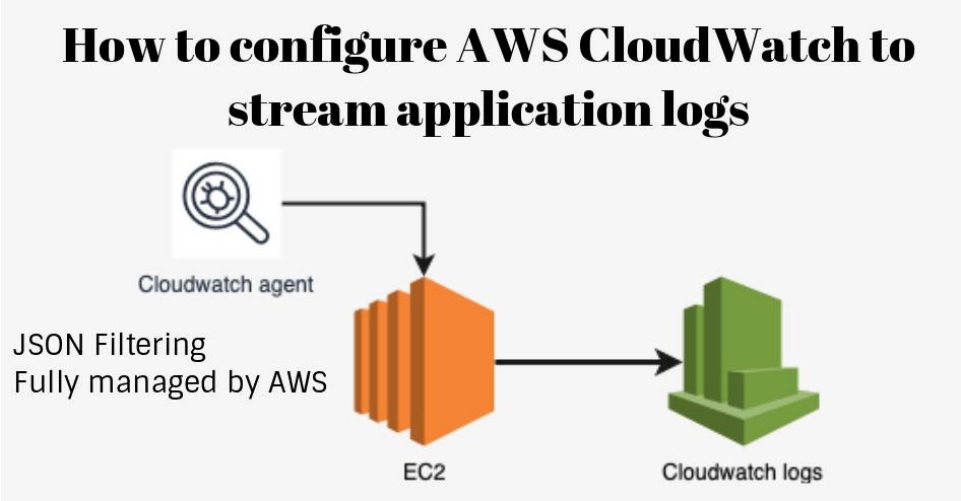


Figure 2.1, AWS EC2, CloudWatch architecture.
https://www.infinitypp.com/amazon-aws/cloudwatch-application-logs/

After the setup of AWS CloudWatch, we then made use of the service AWS Backup. The way we set up our backup is for it to back up the data of the instances every thirty days in the case of a failure or anything like that. Worst case scenario, if anything like this does happen, we can use the backup of the instance to load back up the website and get it to run like nothing ever happened.

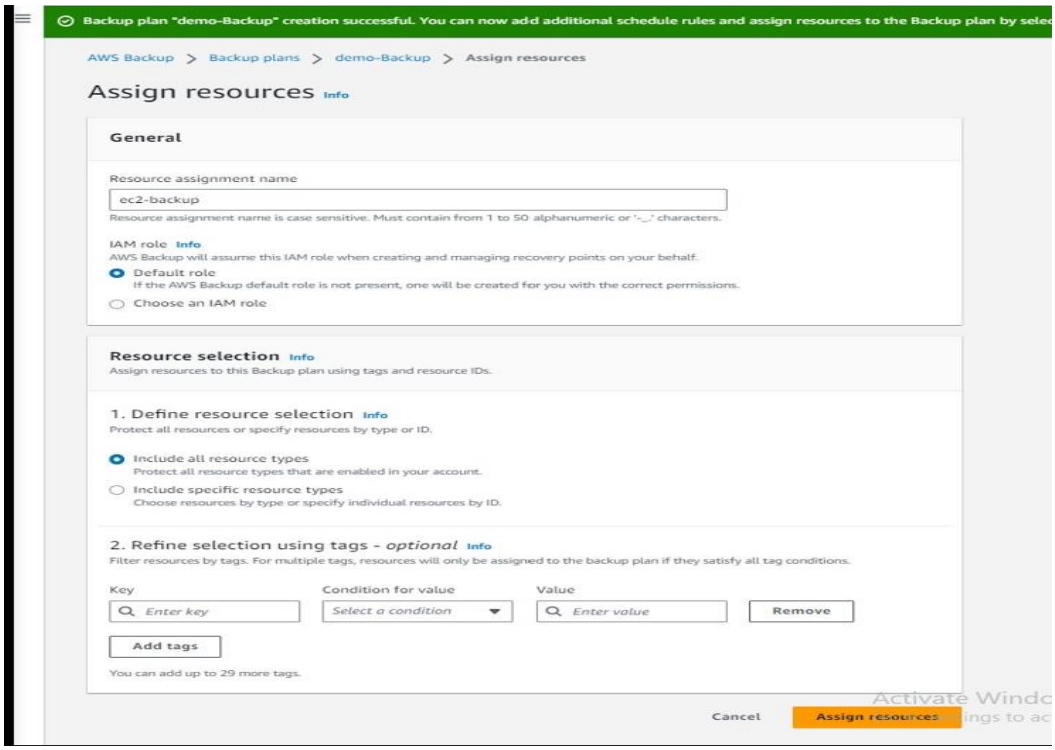


Figure 2.2, back up creation.

Milestone 3: The Final Outcome

The goal of this project was to set up an AWS environment using various services that can host a web application or website. The specific steps we took, such as creating an EC2 instance with Amazon Linux, creating an S3 bucket, adding the s3fullaccess role to the instance profile, and installing the CloudWatch agent, were all necessary components for hosting our web application on AWS.

The EC2 instance provides the computing resources needed to run the application or website, while the S3 bucket provides a scalable and durable storage solution for static files, media, and backups. By adding the s3fullaccess policy to the instance profile, the instance receives permissions for full access to the S3 bucket without needing to manage credentials or access keys. Finally, by installing the CloudWatch agent, you can monitor the performance and health of the EC2 instance and application and troubleshoot issues as they arise. With backup, in the case of a failure of the instance, our instance is backed-up monthly, so we can revert to a previous version of the instance if necessary.

Overall, the goal of this project was to create a basic AWS environment that can host a web application or website, and to ensure that the environment is scalable, reliable, and secure. We were able to accomplish this, as our website was eventually live and interactable, and there were monitoring and backup solutions in place. By leveraging AWS services such as EC2, S3, IAM, and CloudWatch, web applications can be built and deploy your applications or websites more efficiently and with greater flexibility than with traditional on-premises infrastructure.

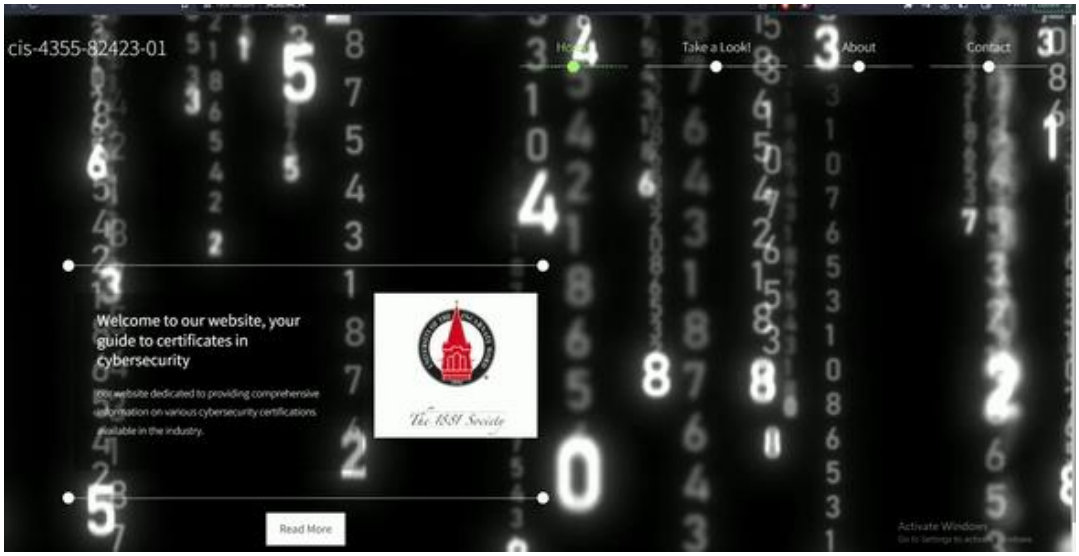


Figure 3.0, live website.

Milestone 4: Architecture

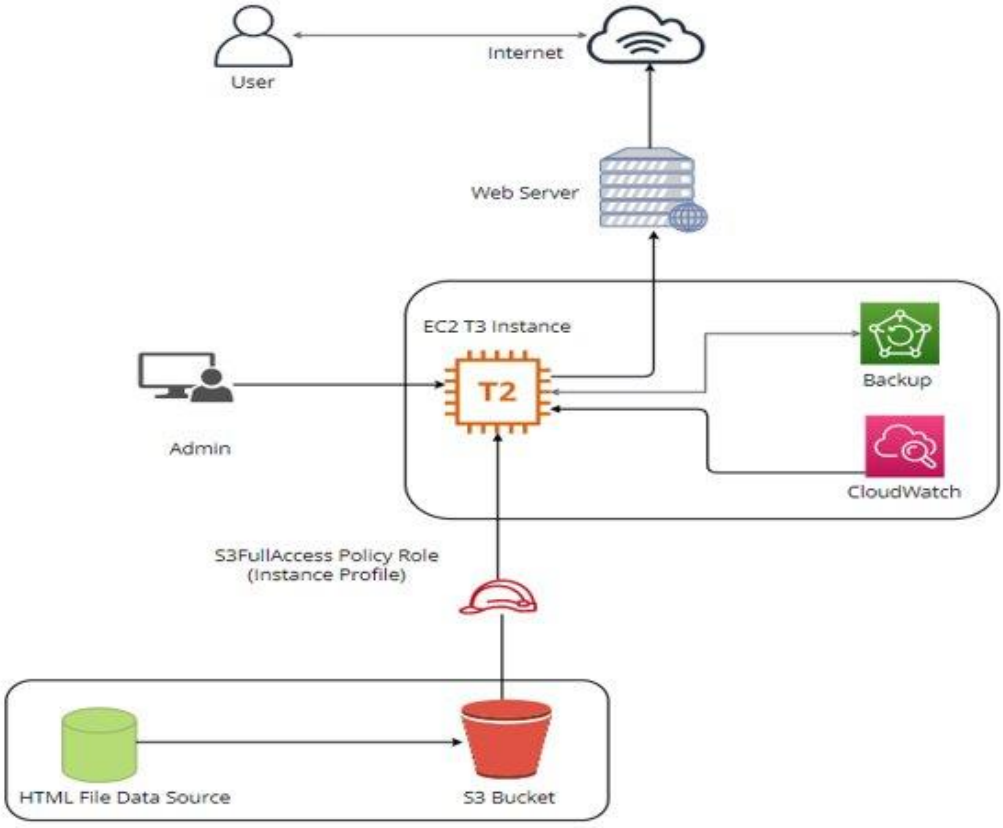


Figure 4.0, architecture of our web application

The architecture of our web application is simplistic but is reliable and works well. Our EC2 instance is the heart of the architecture and keeps the web server up and running no matter if another service decides to fail. The S3 bucket simply serves as data holder and initial entry point for our HTML data to be exported to the instance. Of course, this is only possible with the instance profile with role and S3FullAccess allowing the EC2 instance to retrieve files from the S3 bucket. For web application modification, we could simply replace the files in the bucket with the modifications, and then export and replace the files in the instance. The admin in the diagram refers to the super user on the EC2 system, which allows us to perform functions necessary on the instance such installing and starting the httpd service.

The Backup and CloudWatch services are connected directly to the EC2 instance and have no relation with the S3 bucket nor the instance profile. With this into account, the EC2 instance then starts and hosts the web application, connecting it to the internet, which is accessible at the public ipv4 address, which can then be accessed by users of the site. We did not include the Billing or Cloud Shell services in this diagram. These services did not necessarily hold a key position from an architectural standpoint. AWS billing is simply a

calculated conglomeration of the total for the services used. AWS Cloud Shell was simply used for the execution of certain commands involving the S3, IAM, and EC2 services.

References

- Cloud Computing Services - Amazon Web Services (AWS)
- http://52.55.93.88/
- Github-https://github.com/alqarni360/cyber-security-projects/blob/4f701668c3a673fb970ed5733b61dbbd6cdd5e13/UIW%20Final%20Project.pptx
- https://www.infinitypp.com/amazon-aws/cloudwatch-application-logs/