**Cloud-Based Music Streaming Service with AWS Amplify**

*A Course Project Report Submitted in partial fulfillment of the course requirements for the award of grades in the subject of*

**CLOUD BASED AIML SPECIALITY**

**(22SDCS07A)**

by

**Sujith Bojjawar**

**2210030340**

*Under the esteemed guidance of*

**Ms. P. Sree Lakshmi**

Assistant Professor,

Department of Computer Science and Engineering



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**K L Deemed to be UNIVERSITY**

*Aziznagar, Moinabad, Hyderabad,*

*Telangana, Pincode: 500075*

April 2025

**K L Deemed to be UNIVERSITY**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

*Certificate*

This is Certified that the project entitled **“**Cloud-Based E-commerce Platform with AWS Lambda Integration**”** which is a experimental work carried out by Sujith Bojjawar(2210030340), in partial fulfillment of the course requirements for the award of grades in the subject of  **CLOUD BASED AIML SPECIALITY**, during the year **2024-2025**. The project has been approved as it satisfies the academic requirements.

**Ms.P.Sree Lakshmi Dr. Arpita Gupta**

**Course Coordinator Head of the Department**

**Ms. P. Sree Lakshmi**

**Course Instructor**

**CONTENTS**

Page No.

1. Introduction 4

2. AWS Services Used as part of the project 5

3. Steps involved in solving project problem statement 7

4. Stepwise Screenshots with brief description 10

5. Learning Outcomes 13

6. Conclusion 14

7. References 14

1. **Introduction**

In today’s technologically driven world, cloud computing has revolutionized the way digital services are developed, deployed, and maintained. It enables organizations and developers to scale applications effortlessly, reduce infrastructure costs, and enhance reliability. One of the most impactful uses of cloud platforms is hosting websites and applications without the need for traditional physical servers. Among the cloud service providers, **Amazon Web Services (AWS)** remains a global leader due to its comprehensive suite of services, robust security, scalability, and flexibility. This project focuses on deploying a static website using **Amazon S3 (Simple Storage Service)** and **AWS Amplify**, two fundamental services offered by AWS that simplify and accelerate the deployment of modern web applications.

**Amazon S3** provides object storage with high availability and automatic scalability, allowing users to store and retrieve any amount of data at any time [1]. While its primary purpose is storage, S3 is also equipped to serve as a static web hosting platform. With just a few configurations, a user can enable the **Static Website Hosting** feature, define the root and error documents, and deploy frontend files such as index.html, style.css, and JavaScript assets directly to the web [1], [5]. This reduces the complexity of traditional server-based hosting by removing the need to manage web servers or backend infrastructure. The process includes creating a bucket, uploading files, configuring public-read permissions through **bucket policies**, and enabling website hosting—steps that are straightforward but crucial for successful deployment.

In addition to S3, **AWS Amplify** plays a pivotal role in modern application deployment workflows. It provides developers with tools and services to build, test, and deploy full-stack applications. For this project, AWS Amplify was leveraged for deploying the frontend code stored in the S3 bucket to a custom URL with continuous integration support and a streamlined interface [3]. Amplify simplifies the entire DevOps cycle by allowing integration with code repositories such as GitHub, GitLab, and Bitbucket, or even manual uploads from local machines. Its integration with Amazon S3 ensures that developers can deploy static websites in minutes while benefiting from scalability, global content delivery via AWS CloudFront, and out-of-the-box HTTPS support [3], [6].

A critical component in deploying secure and publicly accessible static websites is the proper configuration of **IAM (Identity and Access Management)** policies. These permissions define what actions are allowed and who can perform them. In this project, custom **bucket policies** were crafted to provide **public-read access**, enabling anonymous users to access static assets like HTML files and images [2]. However, this step had to be carefully executed because misconfigurations could either block public access entirely or expose sensitive data unintentionally. Understanding and applying **least privilege principles** through IAM and bucket policies helped reinforce secure cloud architecture practices [2], [4].

Another key takeaway was the realization of how cloud deployment pipelines can significantly reduce manual errors, increase consistency, and accelerate the development lifecycle. With AWS Amplify, deploying changes became as simple as syncing code or triggering a redeploy from the console. This automation not only improved efficiency but also demonstrated how cloud-native solutions support **agile methodologies** and **continuous delivery** pipelines—a must in modern software engineering environments.

In essence, this project offers a clear perspective on the power of combining AWS services to deliver professional-grade solutions with minimal overhead. It demonstrated how hosting a static website—something that traditionally required configuring servers, domain settings, and file permissions—can be reduced to a few intuitive steps using **Amazon S3** and **AWS Amplify**. The experience gained through this process is not only technically enriching but also aligns with current industry practices where speed, scalability, and reliability are paramount. The practical knowledge obtained serves as a foundational block for exploring more complex cloud-native architectures in the future, such as serverless backends, dynamic web apps, and fully managed CI/CD pipelines.

**2. Aws Services used in the project**

This project extensively uses Amazon Web Services (AWS) to enable seamless hosting, storage, and deployment of a static frontend website. The following AWS services were integrated to build a reliable, scalable, and cost-efficient cloud hosting solution:

**2.1 Amazon S3 (Simple Storage Service)**

Amazon S3 is a highly durable object storage service provided by AWS that supports storing any type of data at scale. In this project, S3 was used as the primary storage layer for the static website files including HTML, CSS, JavaScript, and media assets.

The website files were uploaded to a dedicated S3 bucket, which was then configured for **static website hosting** by enabling public access and assigning a bucket policy that allowed read access to all users. This process aligns with AWS's guidelines for hosting static websites using S3 [1][3]. The reliability and 99.999999999% (11 9's) durability of S3 ensure that the website is always available and securely stored.

**2.2 AWS Amplify**

AWS Amplify is a full-stack development platform specifically designed to streamline frontend and mobile app development. In this project, Amplify was used for **frontend hosting**, which simplifies the deployment of static sites directly from an S3 bucket or a Git repository.

Amplify automatically detects the build artifacts in the project and handles deployment, live preview, and hosting. It supports CI/CD workflows, allowing developers to push updates effortlessly and maintain a consistent release cycle [2][4]. Amplify also handles environment variables, HTTPS management, custom domains, and rollbacks—all essential features in a modern deployment pipeline [6].

**2.3 AWS Identity and Access Management (IAM)**

While IAM was not used directly through custom roles, it plays a vital background role in securely granting access to services like S3 and Amplify. By ensuring that only authorized users and services have the correct permissions, IAM reinforces the principle of **least privilege** and secures the cloud infrastructure used in this project [1].

**2.4 AWS Management Console**

The entire setup was performed through the **AWS Management Console**, which provided an intuitive graphical interface for managing S3 buckets, configuring public access, and deploying the site using Amplify. The console simplifies the otherwise complex cloud interactions and helps visualize the project infrastructure in real-time [5].

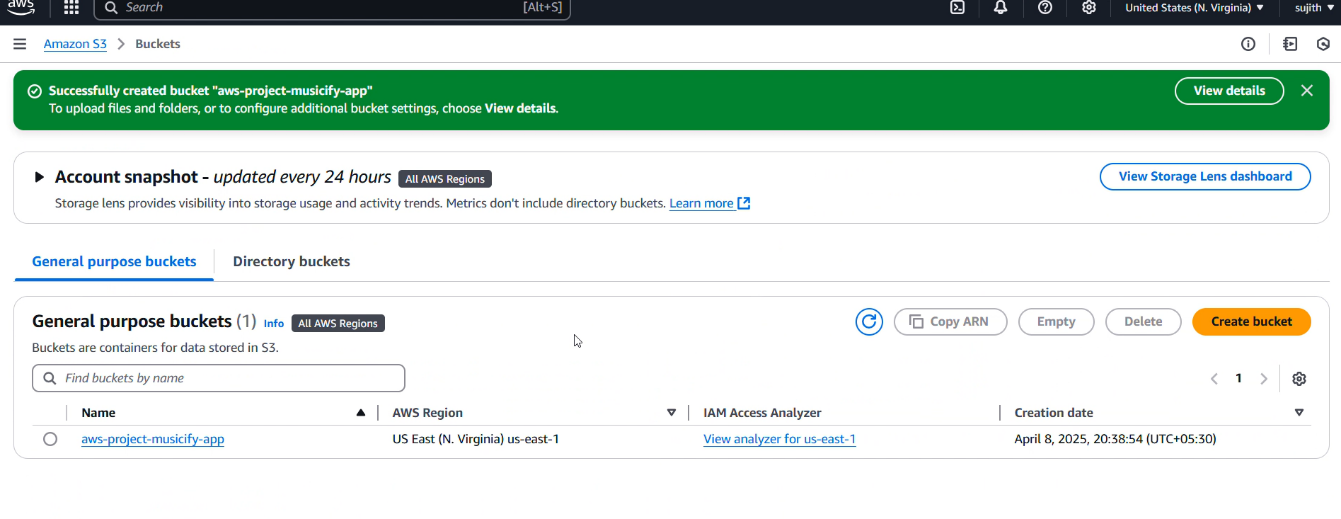
By combining these services, the project was able to demonstrate an end-to-end deployment pipeline for a static website hosted entirely on the cloud, showcasing the power and flexibility of AWS cloud-native solutions [1]–[6].

**3.Step wise Inpmelmentation with brief description**

**Step 1: Create an S3 Bucket**

Begin by navigating to the Amazon S3 service in the AWS Management Console. Click on **"Create Bucket"**, assign a unique name, select your region, and leave the rest as default unless specific configurations are needed.

**Fig-1**



**Step 2: Upload Website Files to the S3 Bucket**

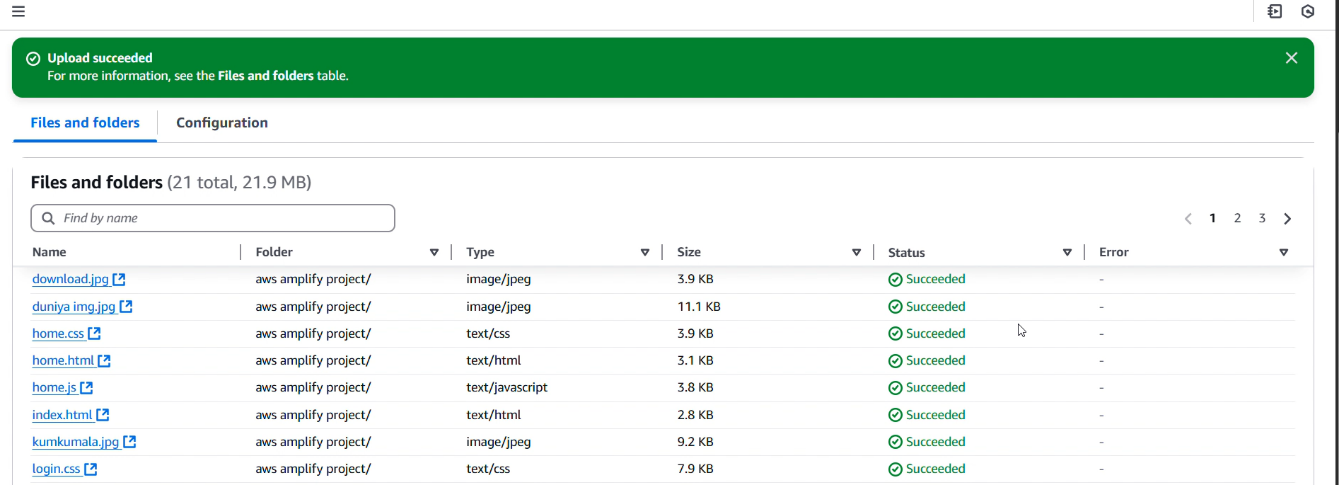
After uploading the required frontend files into the Amazon S3 bucket, the next crucial step is to configure the bucket for **static website hosting**. This transforms the S3 bucket from a simple object storage container into a fully functional static web server capable of serving HTML, CSS, JavaScript, and other static assets directly to users.

To begin, navigate to the **Properties** tab of the created bucket. Scroll down until you locate the section labeled **“Static website hosting.”** By default, this feature is disabled, as Amazon S3 is primarily designed for data storage and not web hosting. However, AWS allows S3 to act as a web server for static content when explicitly enabled.

Click the **“Edit”** button in this section and select **“Enable”**. Once enabled, the interface will prompt you to enter the **index document**—this is the entry point to your website, typically named index.html. Optionally, you can also specify an **error document** like error.html or 404.html to display friendly error messages when invalid URLs are accessed.

Once these fields are filled, click **“Save Changes”** to apply the settings. AWS will now assign a unique endpoint URL (e.g., http://your-bucket-name.s3-website-region.amazonaws.com) which serves as the live link to your static site.

**Fig-2**



**Step 3: Modify Bucket Permissions (Uncheck Block Public Access)**

Go to the **Permissions** tab and click on **"Edit"** under *Block Public Access*. Uncheck all the options to allow public access and acknowledge the warning.

After unblocking public access in the previous step, the next important action is to define a **bucket policy** that explicitly grants public read permissions to the objects stored inside the S3 bucket. This is done through the **Permissions** tab of your S3 bucket.

Scroll down to find the **"Bucket Policy"** section and click on the **“Edit”** or **“Add Policy”** button. The policy editor will open, allowing you to input a custom JSON policy. This policy serves as an access control mechanism that defines who can access what resources in the bucket.

To enable universal read access (i.e., allowing anyone on the internet to access the files in your bucket), paste the following policy and replace your-bucket-name with the actual name of your S3 bucket:

{

"Version": "2012-10-17",

"Statement": [{

"Sid": "PublicReadGetObject",

"Effect": "Allow",

"Principal": "\*",

"Action": "s3:GetObject",

"Resource": "arn:aws:s3:::your-bucket-name/\*"

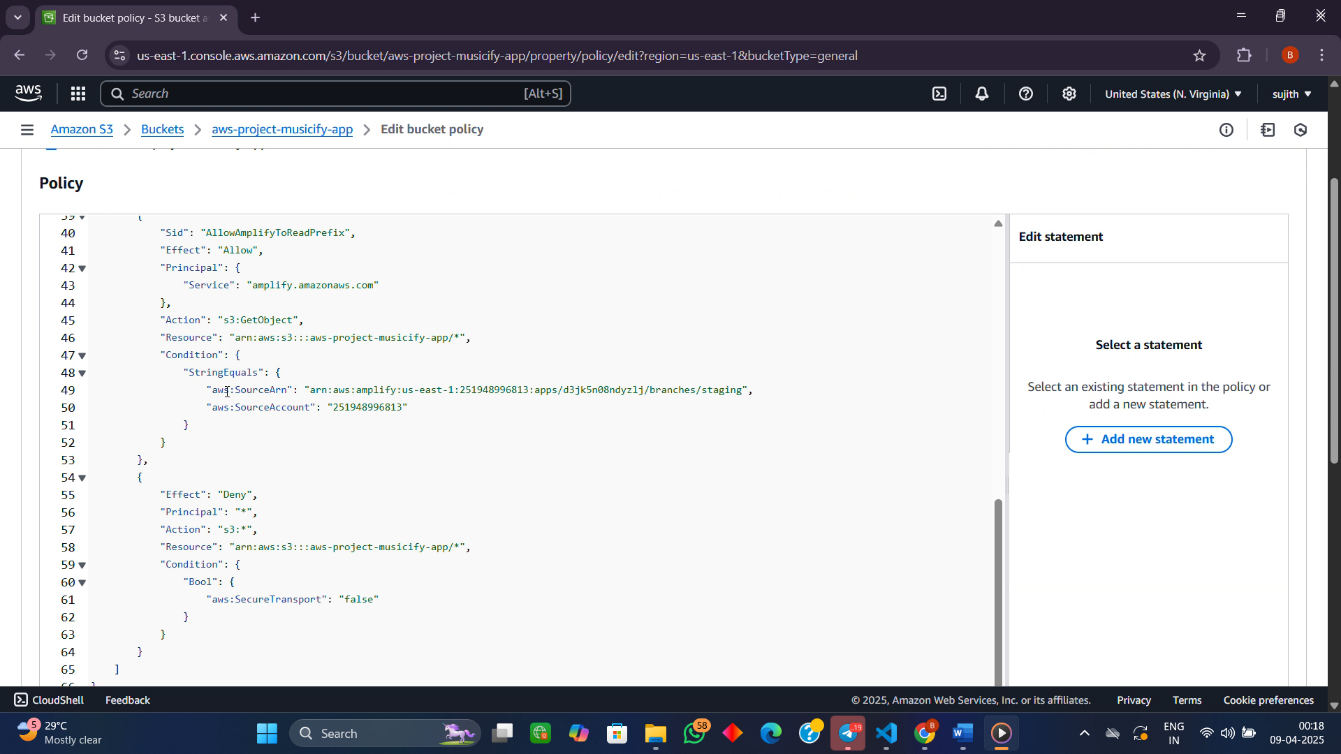
}]

}

This policy permits the s3:GetObject action for all users (Principal: "\*") on all files within the specified bucket. Once entered, click **“Save Changes”**.

This step is crucial to ensure that your static website is publicly viewable. Without this policy, even if static hosting is enabled, users would encounter a “403 Forbidden” error when trying to access your files.

**Fig-3**



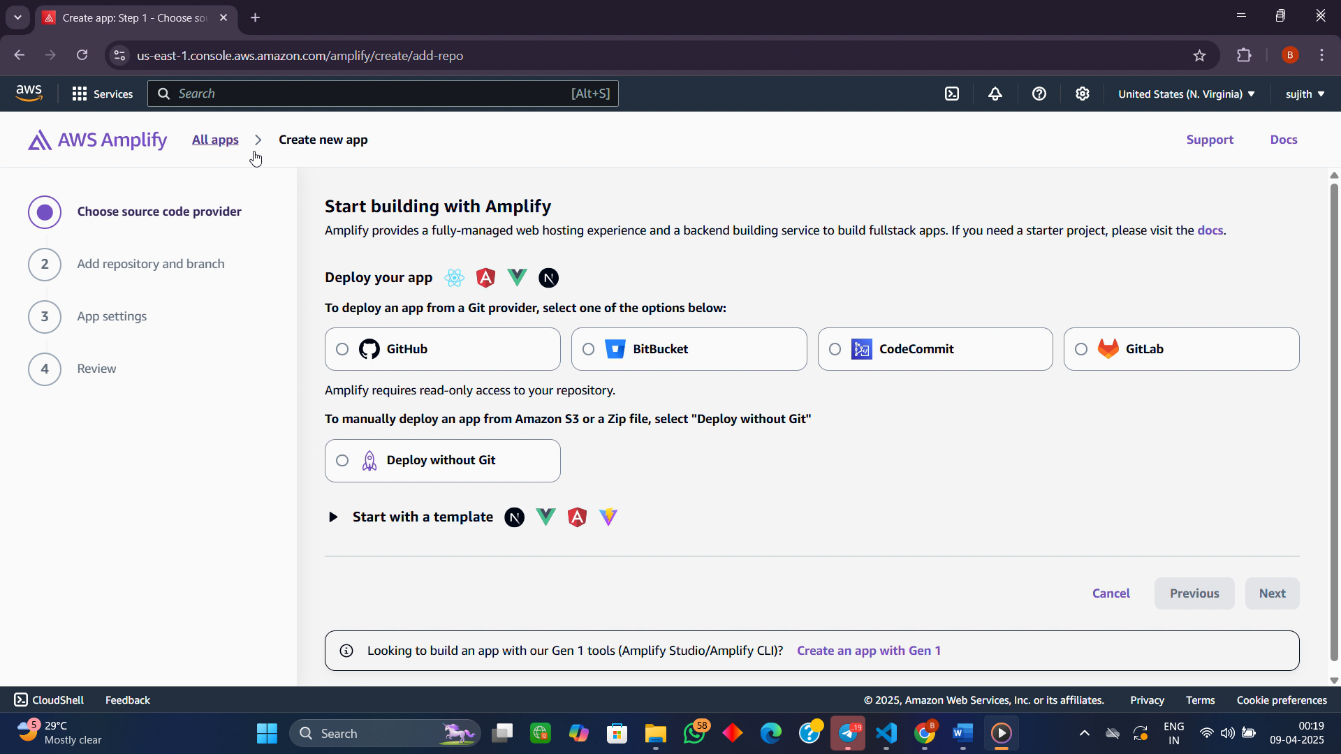
**Step 4: Go to AWS Amplify Console**

Once your S3 bucket is properly set up with all your frontend files (such as index.html, style.css, and JavaScript files), the next crucial step is to enable **static website hosting** within S3. To do this, navigate to the AWS Management Console, go to the **S3 service**, and select your specific bucket. From there, open the **Properties** tab and scroll down until you find the **Static Website Hosting** section. Enable this option and specify index.html as the index document. If desired, you can also set a custom error document like error.html to handle user-side errors gracefully.

Enabling static hosting transforms your S3 bucket into a lightweight web server capable of delivering static content across the internet. However, for a more professional, production-level deployment experience, we prepare to hand this hosting responsibility over to **AWS Amplify**.

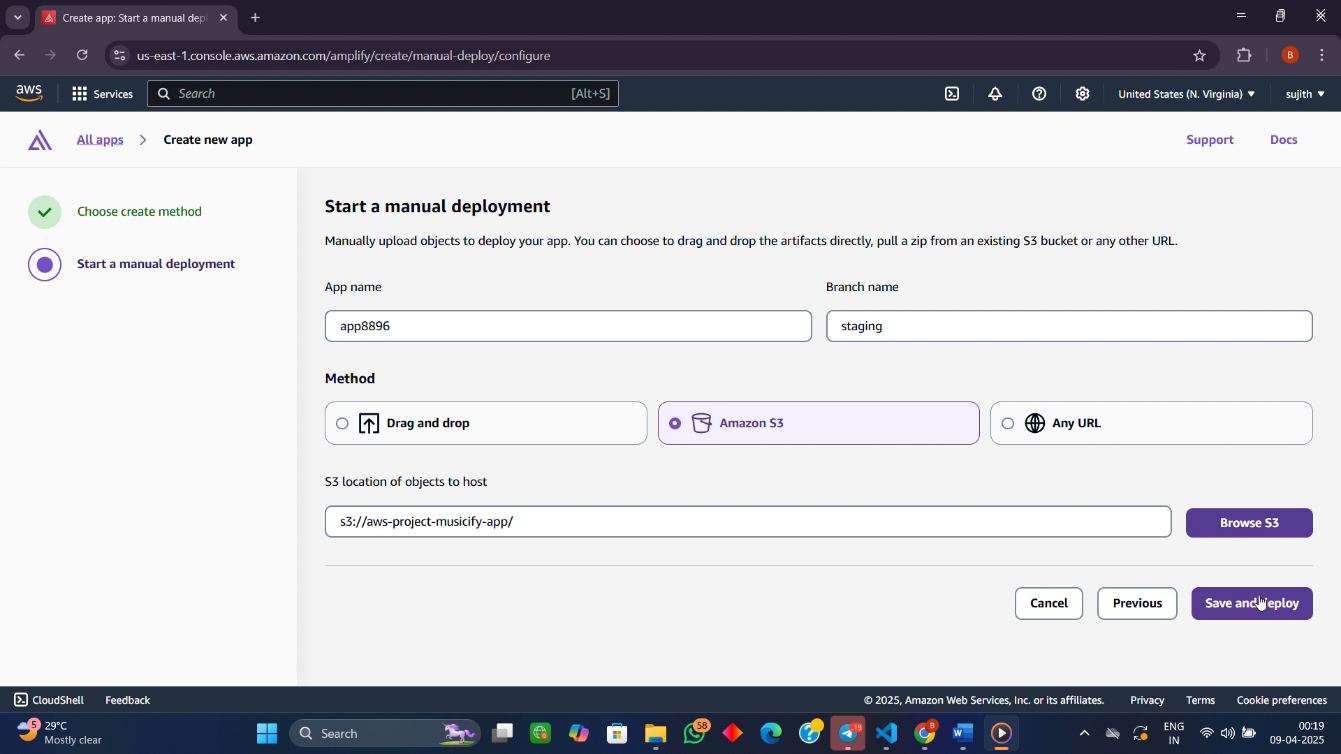
By configuring static website hosting first, Amplify can later detect the correct structure and serve the site efficiently. This step is vital because it lays the groundwork for seamless integration with Amplify’s deployment pipeline, which offers features like custom domains, continuous deployment, and integrated CDN support. With static hosting in place, your project is now ready to move into AWS Amplify for final deployment and public accessibility.

**Fig-4**



**Step 5: Choose S3 as Deployment Source**

**Fig-5**



In the Amplify setup process, choose **"Amazon S3"** as the deployment source. Select the bucket that contains your website files, and proceed with default settings.

**Step 6: Deploy and Access the Website**

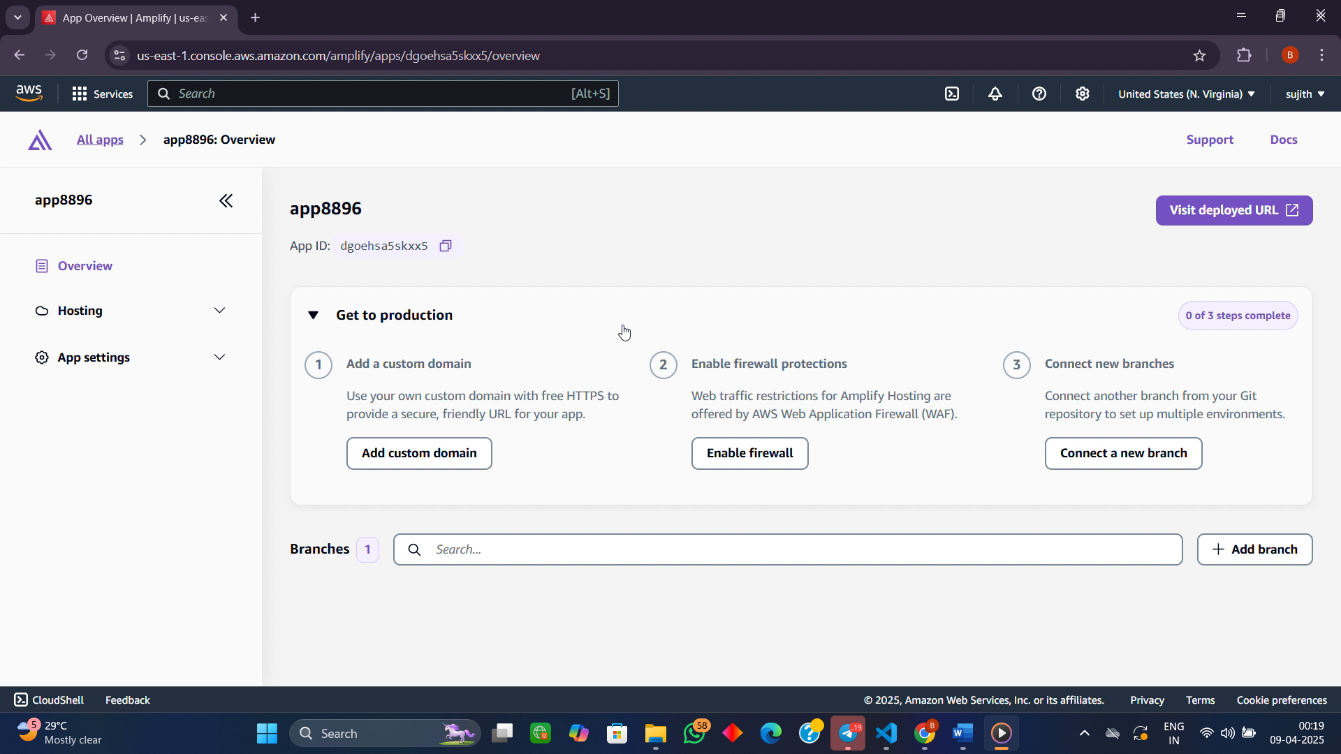
Once all the S3 bucket configurations are properly set—files uploaded, static website hosting enabled, and public-read permissions granted—the next step involves deploying the site using **AWS Amplify** and accessing it via the unique Amplify-hosted URL. AWS Amplify simplifies the deployment process for frontend applications by providing a graphical interface where users can connect their source code repositories (like GitHub) or manually upload build artifacts for deployment.

To start, navigate to the **AWS Amplify Console** in the AWS Management Console. Click on “Get Started” under the **Host web app** section. Choose the **Manual deploy** option since the static files (e.g., HTML, CSS, JS) are already stored in the S3 bucket. Select **Amazon S3** as the source, and from there, choose the appropriate bucket that was configured earlier for static website hosting. Amplify will automatically detect the files and prepare them for deployment.

Once you confirm the setup and click **Deploy**, Amplify will process the build, host the application, and generate a unique domain name under the format https://<app-id>.amplifyapp.com. This URL can be accessed from any browser, allowing users to view the deployed website globally within seconds. Amplify ensures that the site is delivered via a **Content Delivery Network (CDN)**, offering low latency and high performance regardless of user location.

After deployment, AWS Amplify provides a monitoring dashboard where you can track deployment status, view logs, and redeploy new versions of your website as needed. This step completes the development-to-deployment lifecycle, transforming your local project into a live, cloud-hosted application accessible worldwide with just a few clicks—eliminating traditional server setup complexities and enabling a streamlined development experience.

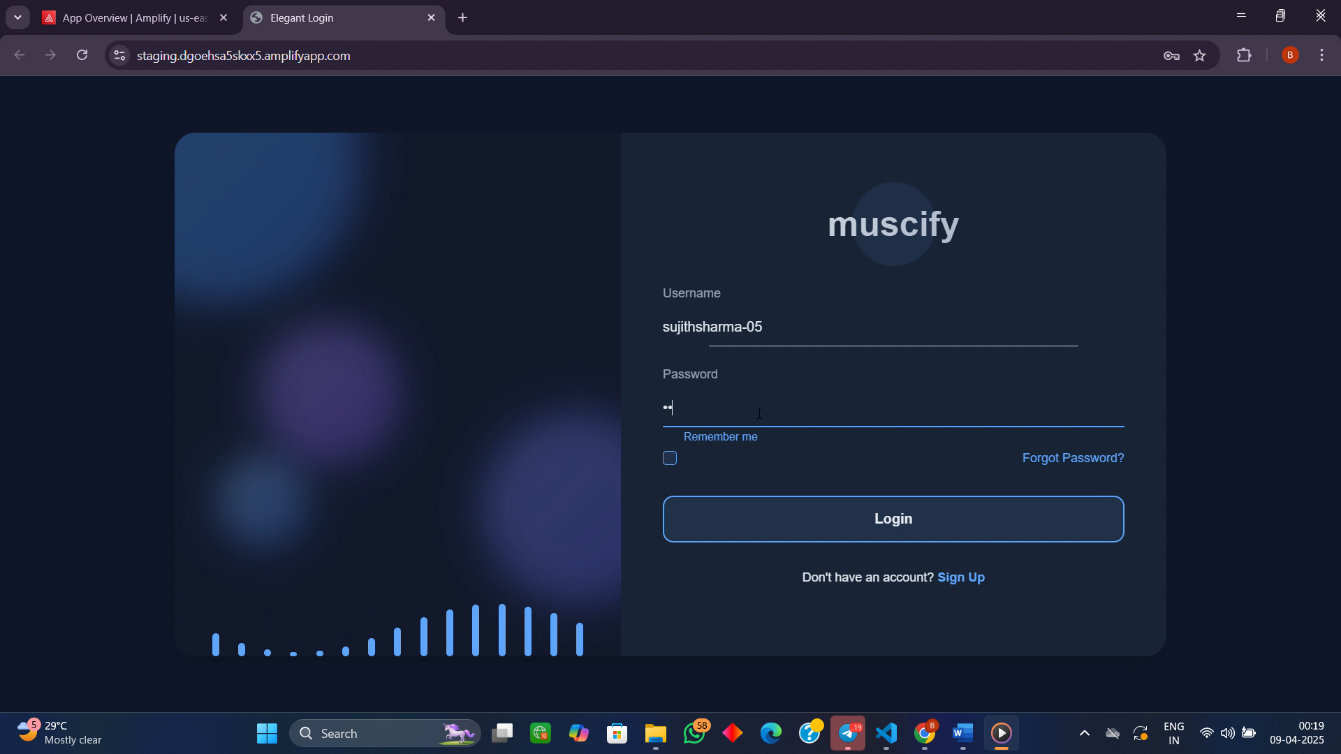
**Fig-6**



After completing the configuration of the S3 bucket and attaching the necessary policies, your static website is now ready for deployment. The final step involves navigating to AWS Amplify and linking the S3 bucket as the hosting source. Once deployed, Amplify generates a public URL through which your website can be accessed globally. This URL provides direct access to the content stored in your S3 bucket, allowing users to view the frontend in any browser.

Now, your website is live and accessible to the public, thanks to the integration between Amazon S3 and Amplify.

**Fig-7**



**4.learning outcomes**

Through the process of deploying a static website using Amazon S3 and AWS Amplify, I gained a comprehensive understanding of cloud hosting and website deployment. The hands-on experience provided me with valuable insights into how cloud services can be utilized for web hosting without the need for traditional web servers. One of the key aspects I learned was how to create and configure Amazon S3 buckets for static website hosting. This experience deepened my understanding of cloud storage solutions, particularly how to enable static website hosting and manage bucket permissions to make the content publicly accessible [1].

Additionally, working with AWS allowed me to dive into configuring **Identity and Access Management (IAM)** policies, which are crucial for securing access to resources. I gained practical experience in setting up public-read access policies to ensure that website content could be accessed globally, while also understanding the importance of security considerations when granting such access [2]. Moreover, integrating AWS Amplify into the workflow streamlined the deployment process, showcasing the power of this service in linking frontend applications to cloud storage, and simplifying the overall deployment pipeline. This step was essential for understanding how cloud services can automate the deployment of web applications, which is critical for scalability and efficiency in modern web development [3].

Finally, this project reinforced my awareness of best practices in cloud security. While enabling public access for the website, I learned to balance accessibility with security, ensuring that the data remains secure while being openly available to users. The lessons learned from configuring secure and efficient cloud environments are applicable to a wide range of future cloud-based projects, solidifying my understanding of the benefits of cloud technologies in hosting static websites and applications [4].

**5.Conclusion**

In conclusion, the process of deploying a static website using Amazon S3 and AWS Amplify has been an insightful and hands-on experience. It provided me with a deeper understanding of cloud-based hosting solutions, particularly for static websites. By configuring Amazon S3 to serve static content and utilizing AWS Amplify for streamlined deployment, I gained practical knowledge of modern web hosting techniques, which are essential for creating efficient and scalable web applications.

This project also highlighted the importance of properly managing access control through IAM policies, ensuring that public access is configured securely. The integration of AWS Amplify allowed me to automate the deployment process, making it easier to launch and manage frontend applications on the cloud. Additionally, the lessons learned from configuring security policies and balancing accessibility with safety are invaluable for future cloud-based web projects.

Overall, this project not only expanded my technical skills in cloud computing but also provided a solid foundation for working with AWS and other cloud platforms in the future. The knowledge gained is directly applicable to real-world scenarios where cloud hosting and deployment play a crucial role in the success of modern web applications.

**6.References**

[1] Amazon Web Services (AWS), "Amazon S3 User Guide," Amazon, 2021. [Online]. Available: <https://docs.aws.amazon.com/s3/index.html>.

[2] Amazon Web Services (AWS), "Managing Access with AWS Identity and Access Management (IAM)," Amazon, 2021. [Online]. Available: <https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html>.

[3] AWS Amplify, "Hosting with AWS Amplify," AWS, 2021. [Online]. Available: https://docs.amplify.aws/hosting.

[4] Amazon Web Services (AWS), "Securing Your Static Website with AWS," AWS, 2021. [Online]. Available: <https://aws.amazon.com/blogs/security/securing-your-static-website-with-aws/>.

[5] Amazon Web Services (AWS), "Overview of Static Website Hosting with Amazon S3," AWS, 2021. [Online]. Available: <https://aws.amazon.com/s3/storage-classes/static-website-hosting/>.

[6] Cloud Academy, "Introduction to AWS Amplify," Cloud Academy, 2021. [Online]. Available: https://cloudacademy.com/learn/aws-amplify/.