Revanth Chowdary Yarlagadda

Revanthcy7598@gmail.com

Abstract

This project compares manual setup and CloudFormation for efficient, scalable static website hosting on AWS.

Hosting a Static Website

AWS

Table of Contents

[Abstract: 1](#_Toc184100245)

[Statement of Need: 2](#_Toc184100246)

[Project Technical Report: 3](#_Toc184100247)

[Manual steup: 3](#_Toc184100248)

[1. Creating s3 bucket: 3](#_Toc184100249)

[2. Setting up the files as objects: 3](#_Toc184100250)

[3. Enabling static website: 4](#_Toc184100251)

[4. Setting permissions of bucket: 5](#_Toc184100252)

[5. Testing the website: 7](#_Toc184100253)

[6. CloudFront distribution: 9](#_Toc184100254)

[7. AWS Budget: 10](#_Toc184100255)

[Cloud Formation Approach: 11](#_Toc184100256)

[1. Creating a stack: 11](#_Toc184100257)

[Outcome of the Project 16](#_Toc184100258)

[Advantages of the Two Approaches 16](#_Toc184100259)

[1. Manual Setup Approach: 16](#_Toc184100260)

[2. Advanced Approach using CloudFormation: 16](#_Toc184100261)

[Conclusion 17](#_Toc184100262)

# Abstract:

This project explores two approaches to deploying a static website on AWS, highlighting both manual setup and an advanced, automated method using AWS CloudFormation. The first approach focuses on a manual setup, where the necessary AWS resources are created and configured individually. This includes setting up an Amazon S3 bucket for static website hosting, managing bucket policies for security, implementing Amazon CloudFront for content distribution, and configuring AWS Budgets to receive cost notifications when thresholds are crossed. The manual approach provides hands-on control over each resource, offering flexibility but requiring more time and effort for management.

The second, advanced approach utilizes AWS CloudFormation to automate the entire deployment process. CloudFormation allows for the creation of a stack, enabling users to define infrastructure as code (IaC) in a template that can be reused to deploy multiple identical setups in the future. This approach eliminates the need to manually configure each AWS resource, as CloudFormation automates the creation and management of the entire environment, enhancing efficiency and scalability. The use of CloudFormation ensures that all resources are deployed in a consistent, repeatable manner, making it an ideal solution for managing infrastructure at scale.

By comparing these two approaches, the project emphasizes the efficiency and scalability benefits of infrastructure automation, while also acknowledging the flexibility provided by manual configuration for smaller or one-time setups.

# Statement of Need:

The statement of need highlights the common challenges faced by businesses and website owners when selecting the appropriate infrastructure for hosting static websites. Traditional approaches, such as utilizing EC2 instances or other server-based solutions, often involve complex configurations, constant maintenance, and ongoing management of server resources. These methods can be resource-intensive and costly, especially for websites with fluctuating or low traffic volumes. For many users, particularly small businesses or individuals without extensive technical expertise, the complexities of maintaining servers and handling infrastructure scalability can be daunting.

Recognizing these challenges, the Static Website Hosting project aims to address these pain points by leveraging Amazon S3 and CloudFront for static website hosting. By using S3, the project eliminates the need for managing servers entirely, offering an easy-to-use and cost-effective solution. S3 provides a scalable platform that automatically handles the storage and delivery of static content, ensuring reliability and reduced operational overhead. The integration of CloudFront further enhances performance by distributing the content globally and ensuring faster load times for users, regardless of their location. This approach simplifies the infrastructure and helps avoid the costs associated with traditional server setups.

Moreover, the use of CloudFormation in this project automates the deployment and management of the entire infrastructure, reducing the need for manual configuration. CloudFormation allows the creation of a repeatable template for the infrastructure, meaning that users can quickly replicate and scale the setup for future projects without having to configure each resource individually. This automation also ensures consistency and accuracy in the environment, enhancing efficiency and minimizing the risk of errors.

Ultimately, the project emphasizes the need for a streamlined, cost-efficient, and easily deployable solution for hosting static websites. By adopting S3, CloudFront, and CloudFormation, it provides an accessible, scalable, and low-maintenance option for website owners, enabling them to focus on content rather than the complexities of server management. This solution offers a perfect balance of flexibility, cost-effectiveness, and scalability, making it an ideal choice for businesses and individuals looking for an efficient method of hosting static websites.

# Project Technical Report:

## Manual steup:

### Creating s3 bucket:

For security reasons, we do not want to provide full access to our S3 bucket to everyone. Instead, we will create a static website and grant access specifically to that website. While we can provide pre-signed URLs for users, this approach would require generating a unique URL for each user, making it more suited for temporary access to specific files or objects within the bucket. However, if we want to avoid granting broader permissions to the entire bucket, creating a static website is a more secure option. This allows us to maintain control over the contents of the bucket while still providing the necessary access to users through the website itself.

We created a s3 bucket with the name: my-s3-static-webiste-bucket

A screenshot of a computer

Description automatically generatedFig: s3 bucket creation

### Setting up the files as objects:

This static website consists of three key components that work together to provide a seamless user experience. Upon visiting the site, the coffee.jpg image is displayed at the start, creating an inviting atmosphere with a coffee-themed visual. This image is called from the welcome.html file, which serves as the main landing page of the website. On successful access, the welcome.html page presents the message, “I really love coffee,” confirming that the website has been loaded successfully and adding a personal touch to the experience. In the event of an error, the error.html file is triggered, displaying a clear **"404 Error"** message, which informs users that the requested page could not be found. These three components work in harmony to ensure the website is functional, visually appealing, and user-friendly.

A screenshot of a computer

Description automatically generated

Fig: uploading objects in bucket

### Enabling static website:

We will click the 'Enable' button to give access to the content of the bucket, while still retaining the ability to set permissions for the actual bucket itself.

A screenshot of a computer

Description automatically generated

Fig: enabled static website hosting

### Setting permissions of bucket:

We can set the permissions of the bucket using a bucket policy. The bucket policy is the first to be accessed when content is being requested. It should include permissions for access as well as any other controls related to IAM roles.

A screenshot of a computer program

Description automatically generated

Fig: Unchecked all public access blocking and bucket policy for permissions of bucket

Other options for permissions include:

* **Access Control List (ACL):** Grants basic read/write permissions to other AWS accounts.
* **Cross-Origin Resource Sharing (CORS):** The CORS configuration, written in JSON, defines a way for client web applications loaded in one domain to interact with resources in a different domain."

For simplicity I didn’t touched any of these permissions but are useful for extra control over the access of the content.

A screenshot of a computer

Description automatically generated

Fig: ACL, CORS

Other options in Amazon S3 include:

* **Lifecycle Rules:** Use lifecycle rules to define actions Amazon S3 should take during an object's lifetime, such as transitioning objects to another storage class, archiving them, or deleting them after a specified period.
* **Replication Rules:** Use replication rules to define options during replication, such as server-side encryption, replica ownership, transitioning replicas to another storage class, and more.
* **Inventory Rules:** Create inventory configurations on a bucket to generate a flat file list of objects and their metadata. These scheduled reports can include all objects in the bucket or be limited to a shared prefix."

For simplicity, we don’t need to use this if we don’t need the advantage of this features and I created this lap for practice, I didn’t used them.

A screenshot of a computer program

Description automatically generated

Fig: lifecycle rules, replication rules, inventory configuration

### Testing the website:

Using the URL under the static website hosting feature, I tried to access the website. It either shows an error message from the error file or calls the object with the welcome file, displaying the content within it, which includes the coffee image and a message.

A hand holding a cup of coffee

Description automatically generated

Fig: successfully loaded the content from the object welcome.html

#### Bucket Features and Configurations:

1. **Tags**: Use bucket tags to track storage costs and organize buckets for easier management and cost allocation.
2. **Intelligent-Tiering Archive Configurations**: Enable objects stored in the Intelligent-Tiering storage class to automatically move to the Archive Access tier or Deep Archive Access tier, which are optimized for objects rarely accessed over long periods.
3. **Server Access Logging**: Log requests made to access your bucket. You can use CloudWatch to monitor the health of server access logs and ensure proper logging functionality.
4. **AWS CloudTrail Data Events**: Configure CloudTrail data events to log Amazon S3 object-level API operations, which you can then view in the CloudTrail console.
5. **Event Notifications**: Set up notifications to alert you when specific events occur in your bucket, helping you stay informed of important actions.
6. **Amazon EventBridge**: Use Amazon EventBridge to build event-driven applications at scale by leveraging S3 event notifications for more advanced event-handling capabilities.
7. **Transfer Acceleration**: Use an accelerated endpoint to enable faster data transfers, enhancing upload and download performance.
8. **Object Lock**: Store objects using a write-once-read-many (WORM) model to prevent deletion or overwriting for a fixed duration or indefinitely. Note that Object Lock only works with versioned buckets.
9. **Requester Pays**: When enabled, the requester is responsible for paying the request and data transfer costs. This also disables anonymous access to the bucket.

**Metrics and Monitoring:**

1. **Bucket Metrics**: Explore various metrics for usage, requests, and data transfer activity within your bucket. These metrics are also available in Amazon CloudWatch for monitoring and analysis.
2. **Storage Class Analysis**: Analyze storage access patterns to determine when objects should transition to the most appropriate storage class, including options like Intelligent-Tiering.
3. **Replication Metrics**: Monitor the status of replication by viewing metrics such as the number and size of objects pending replication, maximum replication time, and any replication failures.
4. **Access Points**: Access points are named network endpoints attached to buckets, simplifying the management of data access at scale. Check if any of the access points grant public or cross-account access to ensure proper security.

### CloudFront distribution:

I created a CloudFront distribution, which is a content delivery network (CDN) that stores cached content at edge locations. This results in much faster access, especially if the origin bucket is in a different region.

A screenshot of a computer

Description automatically generated

Fig: CloudFront for caching at edge location

We need to declare the origin of the CloudFront distribution as the S3 bucket we created, so that the content from the bucket will be cached at the specified edge locations. CloudFront offers pricing classes that allow us to choose regions where the content will be cached. We can select all regions, specific regions, or just North America and Europe. Depending on the geographic location of the users, we can choose the appropriate CDN configuration.

A screenshot of a computer

Description automatically generated

Fig: the origin of the bucket

#### CloudFront Features:

1. **Web Application Firewall (WAF)**: AWS WAF helps protect your application from common web threats and security vulnerabilities. It blocks malicious requests before they reach your web servers, enhancing security. Additionally, you can restrict access based on geographic location, allowing for further control over who can access your content.

### AWS Budget:

We can create budgets to set threshold limits for our accounts. While we can check the costs incurred using Cost Explorer, budgets are particularly useful when we want to set spending limits over a specific period. This ensures that we can monitor and control costs more effectively, preventing unexpected overspending.

A screenshot of a computer

Description automatically generated

Fig: The Budgets

## Cloud Formation Approach:

### Creating a stack:

We need to create a stack in CloudFormation to generate a template that includes all the necessary resources in AWS. Once the stack is created, CloudFormation will automate the process of provisioning and managing these resources. As Infrastructure as Code (IAC), this template can be reused for similar purposes or updated with additional elements as needed. This approach greatly enhances automation and simplifies the process of resource management, ensuring consistency across environments.

A screenshot of a computer

Description automatically generated

Fig: 3 Creating stack

From the figure, we can see there are three options for creating a CloudFormation template.

* 1. The first option is to use an existing template, which can be provided via an AWS S3 URL, uploaded directly from a file, or synchronized with a Git repository.
  2. The second option is to use a sample template that is closest to our use case and edit it as needed.
  3. The final approach is to build a custom template using the Infrastructure Composer.

Once we have the template, we can proceed to the next steps in setting up the necessary permissions and configurations.

I choose the 3rd option.

The Infrastructure Composer offers two ways to create the template with the components we need. The first option is using a canvas with a drag-and-drop method, allowing us to visually match the components. While this approach is intuitive, it can become complex for larger setups. The second option is to write the template manually in either YAML or JSON format, providing more control and flexibility, especially for larger and more intricate configurations.

**Using canvas to create templates:**

A screenshot of a computer

Description automatically generated

**Using code:**

A screenshot of a computer program

Description automatically generated

After successfully adding components in either of the approaches, we can validate the configuration and then generate the template. Once the template is generated, we can set up additional options for the stack. For simplicity, I chose the default options and created the stack. To test, I created only an S3 bucket, and CloudFormation automatically created the S3 bucket for me.

A screenshot of a computer

Description automatically generated

Fig: settings cloudformation permissions and stack failure options

A screenshot of a computer

Description automatically generated

Fig: Advanced options for better control

Once we successfully created the template, we should receive status checks confirming that the template was created and that it worked without any errors.

A screenshot of a computer

Description automatically generated

Fig: Status check

U can update stack with other components too n use that template for future use

A screenshot of a computer

Description automatically generated

Fig: Template

In the tags, we can see that the bucket was created using CloudFormation, and it launched with blocked public access along with a sample policy. I selected these options while creating the template, so next time, it will create the same configuration.

A screenshot of a computer program

Description automatically generated

Fig: verifying the creation

A screenshot of a computer

Description automatically generated

Fig: unblock public access

# Outcome of the Project

The project successfully demonstrated two distinct approaches for hosting a static website on Amazon Web Services (AWS). The first approach involved manually setting up an S3 bucket for static website hosting, configuring the necessary bucket policies, and integrating CloudFront for content delivery. The second approach leveraged CloudFormation to automate the process, enabling future deployments with a reusable template. Through both methods, the website was successfully hosted, optimized for speed and reliability, and monitored for cost and performance.

## Advantages of the Two Approaches

### Manual Setup Approach:

* + **Simplicity for Small-scale Implementations:** Ideal for small projects or one-off setups where automation may not be necessary. It provides a straightforward way to quickly get started with hosting a static website.
  + **Direct Control over Configuration:** This approach gives more granular control over the individual components, such as S3 bucket permissions, CloudFront distribution settings, and budget management.
  + **Cost Monitoring:** By using AWS Budgets and Cost Explorer, users can actively track their spending and ensure they remain within predefined thresholds, preventing unexpected costs.

### Advanced Approach using CloudFormation:

* + **Automation & Efficiency:** CloudFormation automates the entire infrastructure setup process, reducing the time and effort needed for creating and managing resources. Once a template is defined, it can be reused to spin up identical setups quickly and efficiently.
  + **Consistency & Reusability:** The CloudFormation template ensures that each deployment is consistent, with identical settings for S3, CloudFront, and other AWS services, eliminating the potential for manual errors and inconsistencies.
  + **Scalability for Larger Projects:** This approach is ideal for managing large-scale projects or environments where multiple similar setups are required. It reduces the overhead of setting up each resource manually and makes it easier to manage complex infrastructures.
  + **Version Control and Updates:** CloudFormation templates can be version-controlled and updated over time. Changes can be made to the template and then applied to the entire infrastructure without needing to manually reconfigure each resource.

# Conclusion

This project demonstrated two approaches for hosting a static website on AWS: the manual setup and the automated approach using CloudFormation. The manual setup allowed for direct control over resources like S3, CloudFront, and permissions, making it suitable for smaller, simpler projects. However, it required more manual effort and time for each setup.

On the other hand, the CloudFormation approach offered a more efficient, scalable, and automated solution. By defining infrastructure as code, CloudFormation ensured consistency and reduced the chances of human error. It also allowed for easy replication of the setup, making it ideal for larger or growing projects.

In summary, while the manual approach is good for quick, small-scale implementations, the CloudFormation approach is more advantageous for automating and scaling infrastructure with minimal manual intervention. Both approaches highlight the flexibility of AWS, catering to different project sizes and needs.