Project 1: Host a Static Website on S3 with CloudFormation

Detailed Information

Prepared on May 21, 2025, at 01:45 PM IST

1 Main Goal

The goal of this project is to set up a static website on an AWS S3 bucket using a CloudFormation template. A static website is a simple webpage with fixed content, like HTML files, that does not change unless updated manually. This project aims to:

- Use S3 as a web hosting service instead of just a storage space.
- Automate the creation of the S3 bucket and its configuration using CloudFormation.
- Make the website publicly accessible so anyone can view it on the internet.

This project is beginner-friendly and demonstrates how to use AWS services efficiently for hosting simple websites.

2 Main Use

Hosting a static website on an S3 bucket has several practical uses:

- **Cost-Effective Web Hosting**: S3 is a cheap way to host simple websites, costing just a few cents a month for small sites. It is perfect for personal portfolios, resumes, company landing pages, or event pages.
- **Creative Use of Storage**: S3 is mainly a storage service for files like photos or backups, but its static website hosting feature lets it serve HTML pages to visitors, making it a versatile tool.
- **Fast and Reliable**: S3 uses AWS's global network, so the website loads quickly for visitors and is available 24/7 without crashing, even with many visitors.

This project shows how to use a storage service like S3 for web hosting, which is a smart and affordable solution for simple websites.

3 Steps in CloudFormation

This section outlines the entire process of using CloudFormation to set up the static website, from uploading the `.yaml` file to testing the website. These steps are performed within the AWS CloudFormation service and its interface.

1. **Prepare the CloudFormation Template**: First, create a `.yaml` file that defines the resources needed for the static website. The complete template is provided in Section 5. This file includes:

   - A parameter for the S3 bucket name.
   - An S3 bucket resource configured for static website hosting.
   - Ownership controls to disable legacy ACLs.
   - Public access settings to allow a public bucket policy.
   - A bucket policy to make objects publicly readable.
   - Outputs for the website URL and bucket name.

2. **Upload the `.yaml` File to CloudFormation**:
   - Log in to the AWS Management Console.
   - Go to the CloudFormation service.
   - Click `Create stack` and select `With new resources (standard)`.
   - Under `Specify template`, choose `Upload a template file`.
   - Click `Choose file` and upload the `static-website-s3-updated-v2.yaml` file (or the latest version of the template you saved).
   - Click `Next`.

3. **Specify Stack Details and Parameters**:
   - `Stack name`: Enter a name for the stack, such as `StaticWebsiteStackV3`.
   - `Parameters`: The template requires a bucket name:
     - For `BucketName`, enter a unique, lowercase name, such as `my-static-website-2025-tha`
     - The name must be 3–63 characters long, use only lowercase letters, numbers, hyphens, or dots, and be globally unique.
   - Click `Next`.

4. **Configure Stack Options (Optional)**:
   - On the `Configure stack options` page, leave the defaults as they are.
   - Optionally, add tags (e.g., `Project: StaticWebsite`) for organization.
   - Click `Next`.

5. **Review and Create the Stack**:
   - On the `Review` page, verify the stack name and bucket name.
   - Under `Capabilities`, check the box `I acknowledge that AWS CloudFormation might create IAM resources with custom names` (even though this template does not use IAM).
   - Click `Create stack`.
   - Wait for the stack to reach the `CREATE_COMPLETE` status, which typically takes 1–2 minutes.

6. **Get the Website URL from Outputs**:
   - Once the stack is created, go to the `Outputs` tab of the stack.

- Find the key `WebsiteURL`. The value will be the URL of the website, such as `http://my-static-website-2025-tharun.s3-website-us-east-1.amazonaws.com`.
- Copy this URL for testing.
- The `BucketName` output (e.g., `my-static-website-2025-tharun`) confirms the bucket name for the next step.

7. **Upload an HTML File to the S3 Bucket**:
   - Go to the S3 console in AWS.
   - Find the bucket using the `BucketName` from the CloudFormation outputs (e.g., `my-static-website-2025-tharun`).
   - Click on the bucket name, then click `Upload`.
   - Upload an HTML file, such as `indexxx.html`, which was provided earlier:
     - Example content of `indexxx.html`:
       ```
       <!DOCTYPE html>
       <html>
       <head>
         <title>My Static Website - Page 2</title>
       </head>
       <body>
         <h1>Welcome to My Static Website - Page 2!</h1>
         <p>This is a second page (indexxx.html) hosted on S3 using CloudFormat
         <p>Deployed on: May 21, 2025, at 12:50 PM IST</p>
       </body>
       </html>
       ```
   - Click `Upload` to add the file to the bucket.

8. **Test the Website Using the CloudFormation Output URL**:
   - Open a web browser.
   - Paste the `WebsiteURL` from the CloudFormation outputs (e.g., `http://my-static-website-`
   - Since the bucket is configured to use `index.html` as the default page, you need to specify `indexxx.html` in the URL to test this specific file: `http://my-static-website-2C`
   - You should see the webpage with the content of `indexxx.html`, confirming the website is working.

These steps cover the entire process within CloudFormation, from uploading the template to using the outputs to test the website.

4 Why It Is Different from the Manual Process

Using CloudFormation to set up the static website offers several advantages over doing it manually in the AWS console:

- **Automation Saves Time**: Manually, you would need to create the bucket, enable website hosting, set a bucket policy, and adjust public access settings, which takes 5–10 minutes of clicking through the console. CloudFormation does all this in one go in 1–2 minutes.
- **Consistency Reduces Mistakes**: Manually, you might forget a step, like enabling website hosting or setting the right policy, causing errors. CloudFormation follows the template exactly every time, so there are fewer mistakes.
- **Reusability for Multiple Websites**: Manually, creating another website means repeating all the steps from scratch. With CloudFormation, you can reuse the same template by changing the bucket name, making it quick to set up more websites.
- **Easy to Update**: Manually, changing settings (like the bucket policy) requires going back to the console. With CloudFormation, you update the template and redeploy the stack, and it handles the changes for you.
- **Easy to Delete**: Manually, deleting the website means removing the bucket and policies by hand, and you might miss something. CloudFormation deletes everything with one click when you delete the stack, ensuring no leftover resources cost money.

In short, CloudFormation makes the process faster, more reliable, and easier to manage compared to doing it manually.

5 Complete CloudFormation YAML Code

Below is the complete `.yaml` code used for this project. This template was iteratively updated to address issues like S3 bucket ACL conflicts and Block Public Access settings, resulting in the final version shown here.

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Description: CloudFormation template to host a static website on S3

Parameters:
  BucketName:
    Type: String
    Description: Globally unique name for the S3 bucket (lowercase only)
    MinLength: 3
    MaxLength: 63
    AllowedPattern: '^[a-z0-9][a-z0-9.-]*[a-z0-9]$'
    ConstraintDescription: Bucket name must be 3-63 characters long, lowercase letters, numbers, hyphens, or dots only, and must be globally unique

Resources:
  WebsiteBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Ref BucketName
      WebsiteConfiguration:
        IndexDocument: index.html
        ErrorDocument: error.html
      OwnershipControls:
        Rules:
          - ObjectOwnership: BucketOwnerEnforced
      PublicAccessBlockConfiguration:
        BlockPublicAcls: false
        IgnorePublicAcls: false
        BlockPublicPolicy: false
        RestrictPublicBuckets: false
      Tags:
        - Key: Name
          Value: StaticWebsiteBucket

  BucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref WebsiteBucket
      PolicyDocument:
        Statement:
          - Effect: Allow
            Principal: '*'
            Action: 's3:GetObject'
            Resource: !Sub 'arn:aws:s3:::${WebsiteBucket}/*'

Outputs:
  WebsiteURL:
    Description: URL of the static website
    Value: !GetAtt WebsiteBucket.WebsiteURL
  BucketName:
    Description: Name of the S3 bucket
    Value: !Ref WebsiteBucket
```

This template defines all the necessary resources and configurations to host a static website on S3, and it can be reused for similar projects by changing the `BucketName` parameter.