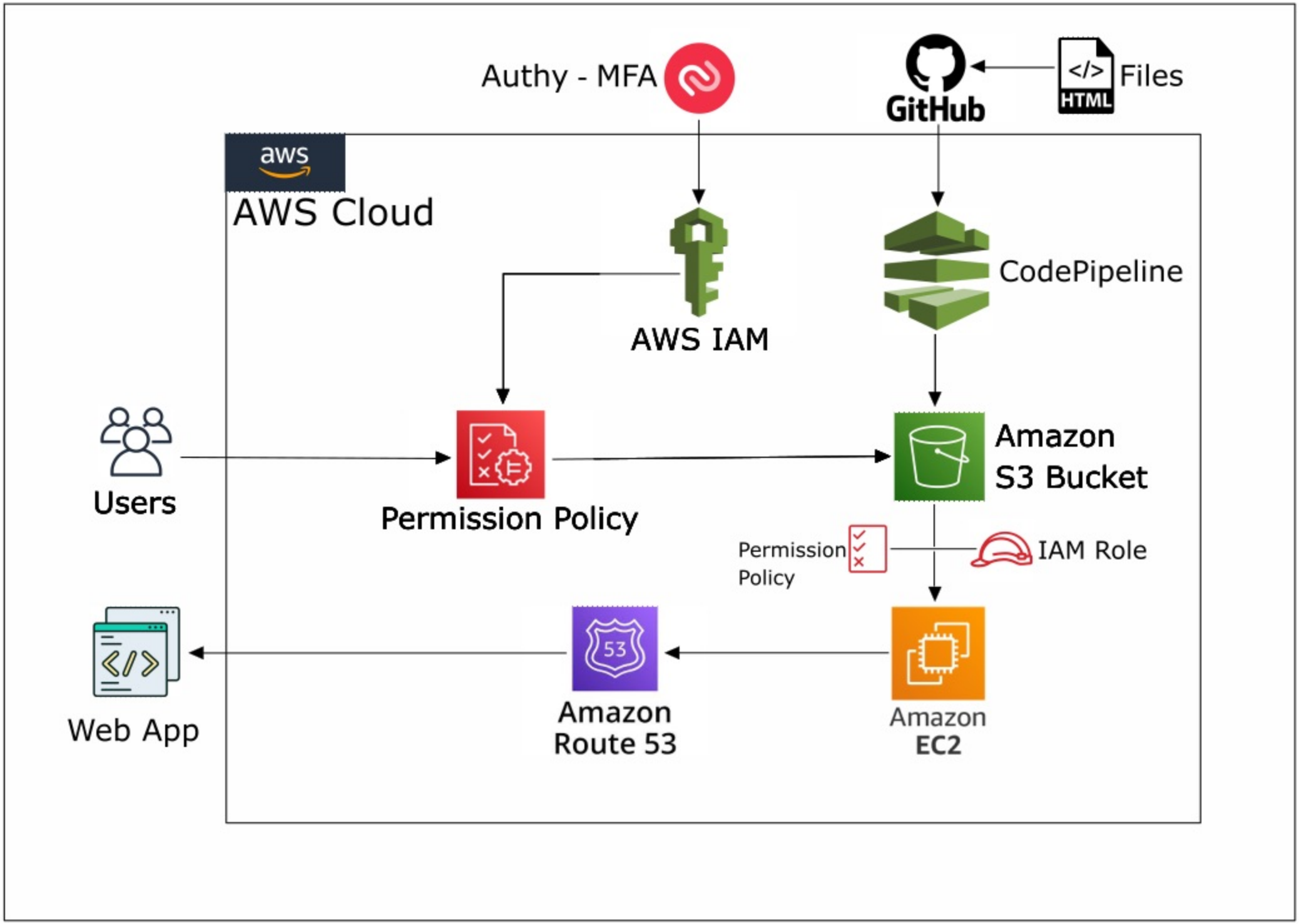
Authy - MFA
GitHub
Files
HTML
aws
AWS Cloud
AWS IAM
CodePipeline
Users
Permission Policy
Amazon
S3 Bucket
Permission
Policy
IAM Role
Web App
53
Amazon
Route 53
Amazon
EC2

Project: Static Website Deployment on AWS S3 with IAM Role Management, EC2 Integration, Route 53, and Automated Access Control

This project involved deploying a static website on Amazon Web Services (AWS) using an S3 bucket, integrated with an EC2 instance, and managed through Route 53. The website files, including HTML, CSS, and JavaScript, were hosted on an S3 bucket, leveraging S3's ability to serve static content directly over the web.

Key components of the project include:

1. S3 Bucket Setup:
- Created an S3 bucket specifically for hosting the static website.
- Configured the bucket to enable static website hosting, setting the index document.
- Uploaded website files to the S3 bucket.

2. IAM Roles and Permissions:
- Defined IAM roles and policies to manage access to the S3 bucket.
- Created a bucket policy to allow public read access to the website content while securing non-public resources.
- Configured IAM users and roles to control who can upload, modify, or delete content in the S3 bucket.

3. Security Measures:
- Implemented best practices for securing the S3 bucket by enabling bucket versioning, logging, and encryption.
- Used IAM policies to restrict access to specific users and services, ensuring that only authorized personnel can make changes to the bucket contents.
- Integrated Authy for Multi-Factor Authentication (MFA) to add an extra layer of security for accessing AWS services.

4. Automated Deployment:
- Utilized GitHub for version control and collaborative development.
- Employed AWS CodePipeline to automate the deployment process, ensuring continuous delivery of updates to the S3 bucket and EC2 instance.

5. EC2 and Route 53 Integration:
- Configured an EC2 instance to serve dynamic content in conjunction with the static files hosted on S3.
- Set up Route 53 to manage DNS routing, ensuring seamless access to the website through a custom domain name.
- Integrated the EC2 instance with S3 to fetch and serve static content as needed.

Outcome:
The project successfully demonstrated how to deploy a static website on AWS S3 with secure access management using IAM, integrated with EC2 and managed through Route 53. The setup also showcased automated deployment using GitHub and CodePipeline, providing a scalable, cost-effective solution for hosting static web content with built-in security, access control, and automation.