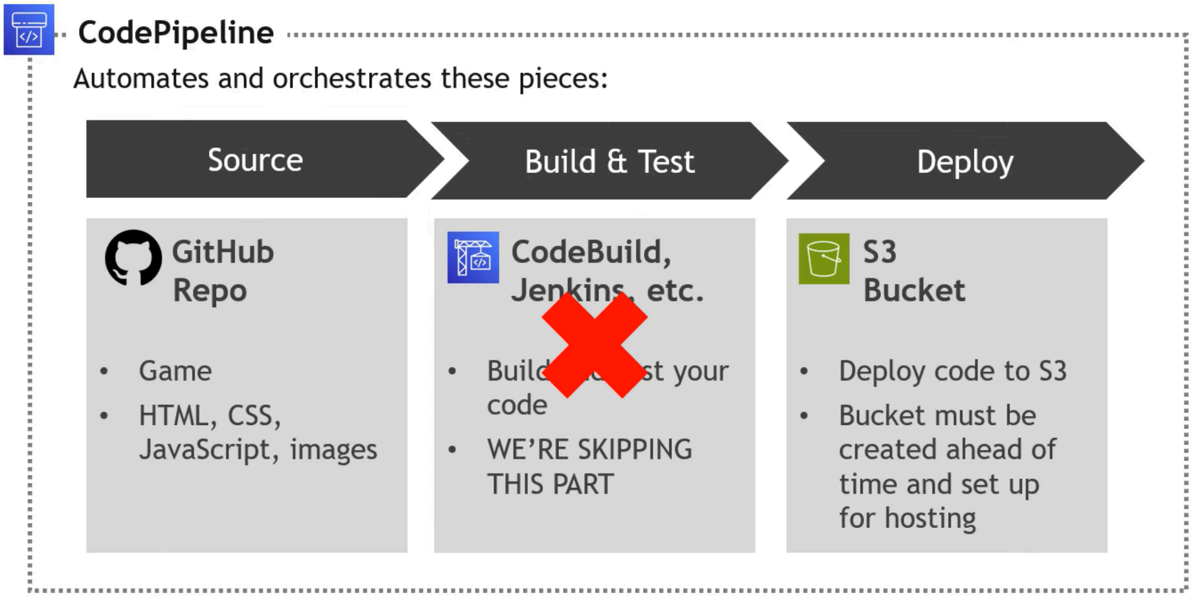
Project Documentation: AWS Continuous Deployment Pipeline

1. Project Overview

Project Title: AWS Meme Matching Game Continuous Deployment Pipeline

Introduction: This project involves building and deploying a simple meme-matching memory card game using AWS services. The core of the project is setting up a continuous deployment pipeline with AWS CodePipeline to automatically pull code from GitHub and deploy it to an S3 bucket configured for static website hosting. The game is built using HTML, CSS, and JavaScript, with no server-side code.


CodePipeline

Automates and orchestrates these pieces:

Source → Build & Test → Deploy

GitHub Repo
- Game
- HTML, CSS, JavaScript, images

CodeBuild, Jenkins, etc.
- Build and test your code
- WE'RE SKIPPING THIS PART

S3 Bucket
- Deploy code to S3
- Bucket must be created ahead of time and set up for hosting

Project Workflow

Video Reference: AWS Project: Build a Game with a Continuous Deployment Pipeline from GitHub to S3

2. Tools and Technologies

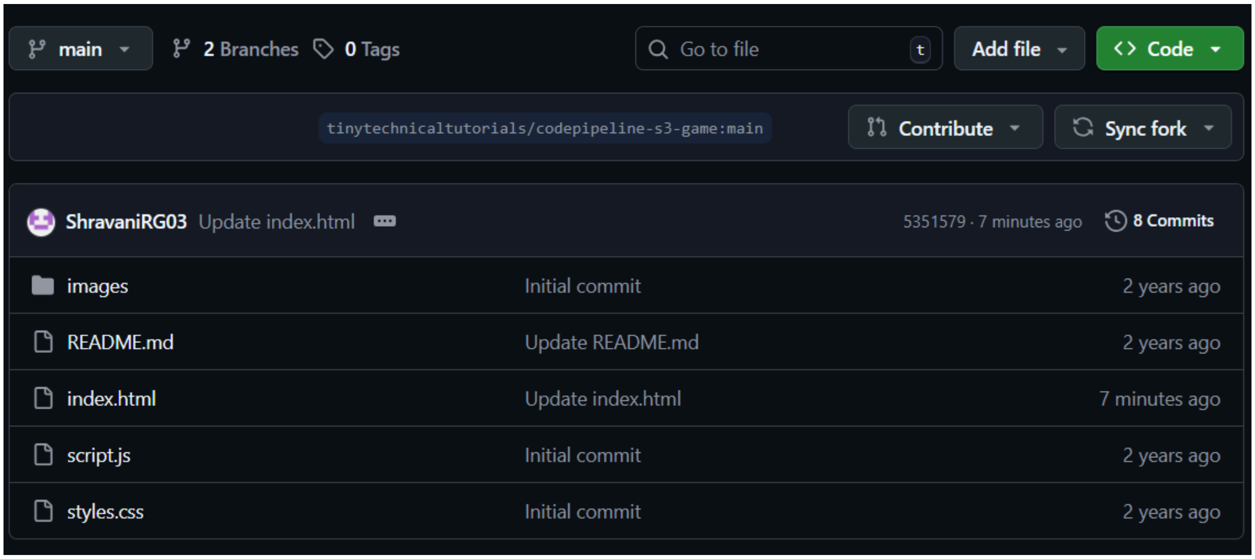
Version Control: GitHub
Cloud Services: Amazon Web Services (AWS)
CI/CD: AWS CodePipeline
Storage: AWS S3 (for static website hosting)
Web Technologies:

HTML, CSS, JavaScript

3. Implementation Steps

Step 3.1: Getting the Project Code from GitHub

Fork the Repository: Start by forking the provided GitHub repository (codepipeline-s3-game) to your own GitHub account. This creates a copy that you can modify and use for your pipeline.



tinytechnicaltutorials/codepipeline-s3-game:main

ShravaniRG03 Update index.html 5351579 · 7 minutes ago 8 Commits

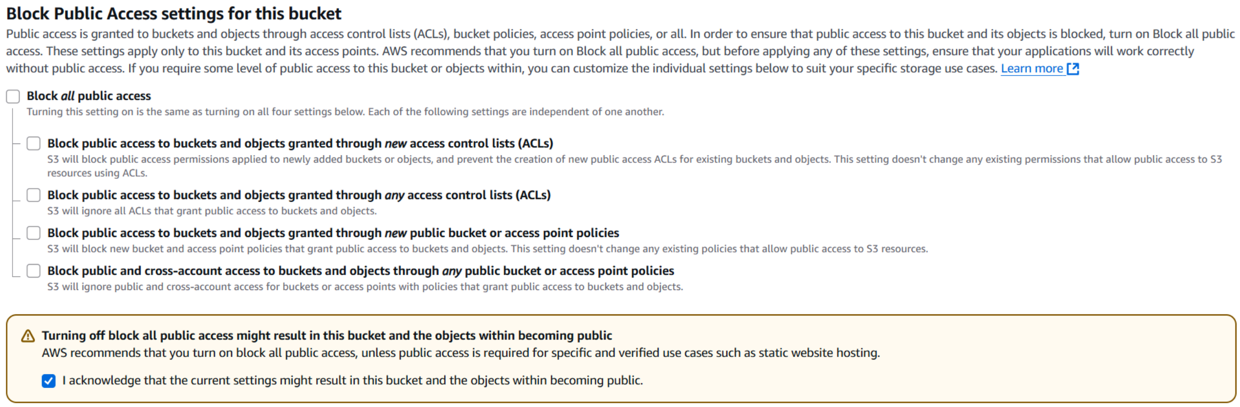
images	Initial commit	2 years ago
README.md	Update README.md	2 years ago
index.html	Update index.html	7 minutes ago
script.js	Initial commit	2 years ago
styles.css	Initial commit	2 years ago

Forked repository

Step 3.2: Creating and Configuring the S3 Bucket

Create the Bucket: In the AWS Management Console, navigate to the S3 service and create a new bucket. Remember that the bucket name must be globally unique.

Disable Public Access: Uncheck the "Block all public access" setting, as the website needs to be publicly available. Acknowledge the warning.



Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

☐ **Block *all* public access**
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

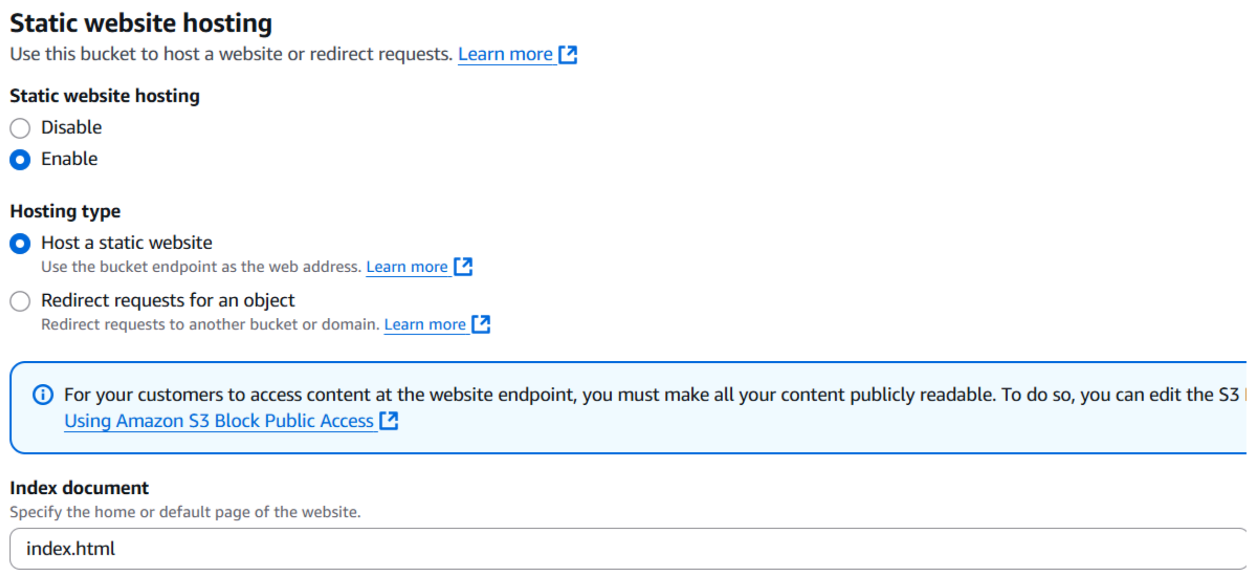
- ☐ **Block public access to buckets and objects granted through *new* access control lists (ACLs)**
  S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- ☐ **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
  S3 will ignore all ACLs that grant public access to buckets and objects.
- ☐ **Block public access to buckets and objects granted through *new* public bucket or access point policies**
  S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- ☐ **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
  S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

⚠ **Turning off block all public access might result in this bucket and the objects within becoming public**
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

☑ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

S3 Bucket configurations

Enable Static Website Hosting: Go to the bucket's "Properties" tab, scroll down to "Static website hosting," click Edit, and enable it. Set index.html as the index document.



Static website hosting

Use this bucket to host a website or redirect requests. Learn more

Static website hosting

○ Disable

● Enable

Hosting type

● Host a static website

Use the bucket endpoint as the web address. Learn more

○ Redirect requests for an object

Redirect requests to another bucket or domain. Learn more

ⓘ For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3

Using Amazon S3 Block Public Access

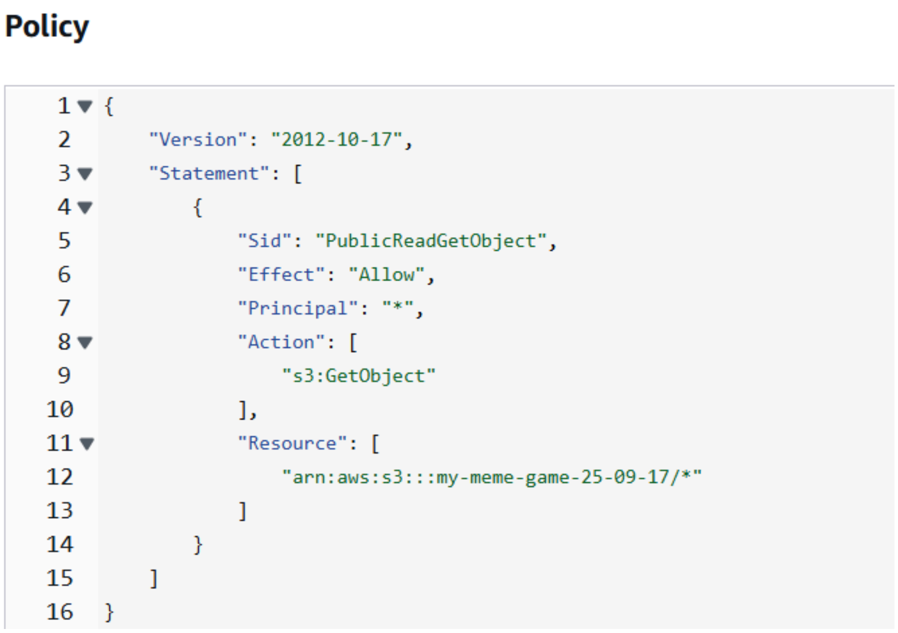
Index document

Specify the home or default page of the website.

index.html

Static website hosting configurations

Add a Bucket Policy: Navigate to the "Permissions" tab and add a bucket policy to allow public read access to the objects in the bucket. The policy should grant the s3:GetObject action to all principals (*) on your bucket's resources.



Policy

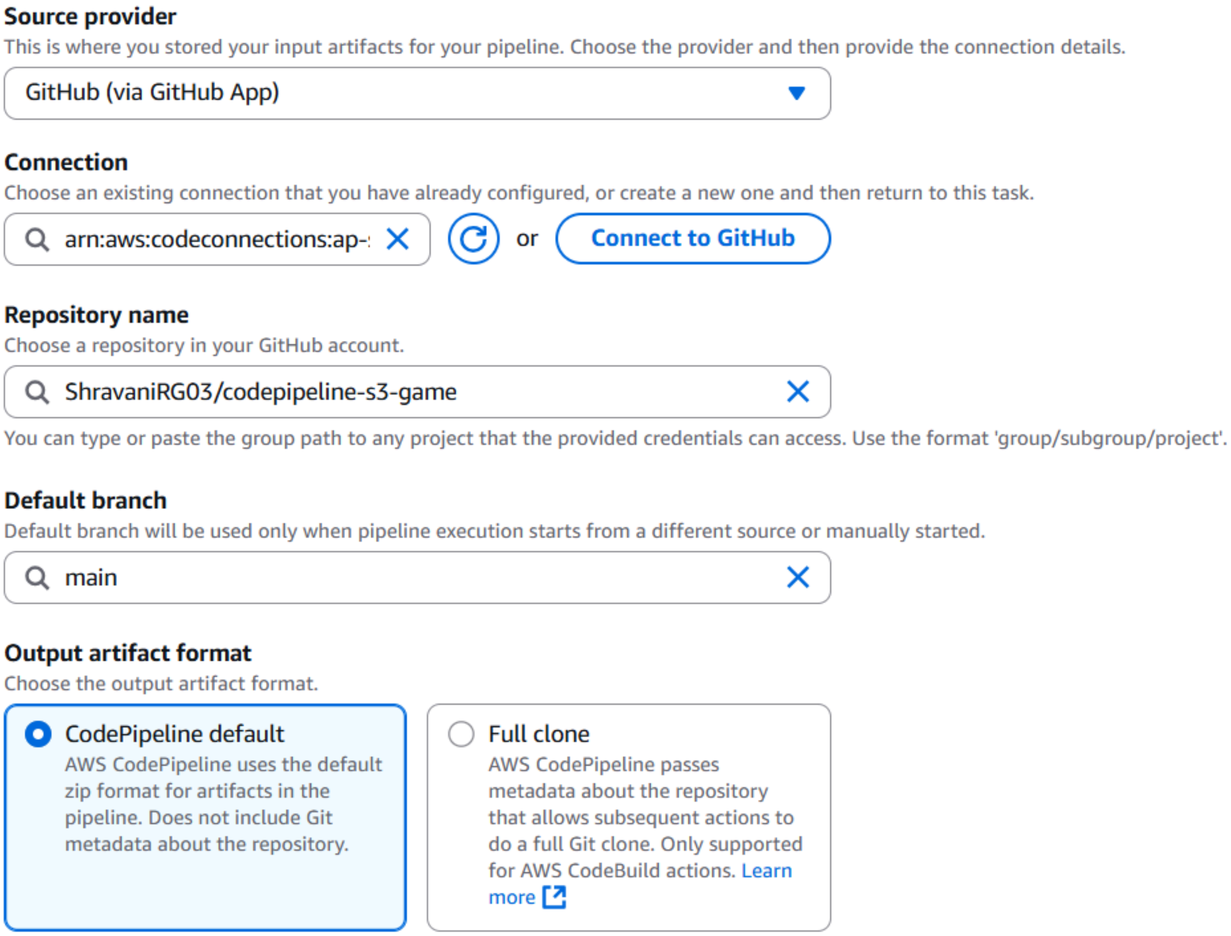
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::my-meme-game-25-09-17/*"
            ]
        }
    ]
}
```

Bucket policy

Step 3.3: Creating the AWS CodePipeline

Create a New Pipeline: Go to the AWS CodePipeline service and click Create pipeline. Give the pipeline a name and create a new service role.

Configure the Source Stage: Select GitHub (Version 2) as the source provider. Connect to your GitHub account and select the repository you forked earlier. Set the detection mode to "GitHub webhooks" to trigger the pipeline on every commit.



Source provider

This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

GitHub (via GitHub App)

Connection

Choose an existing connection that you have already configured, or create a new one and then return to this task.

arn:aws:codeconnections:ap-:

Repository name

Choose a repository in your GitHub account.

ShravaniRG03/codepipeline-s3-game

You can type or paste the group path to any project that the provided credentials can access. Use the format 'group/subgroup/project'.

Default branch

Default branch will be used only when pipeline execution starts from a different source or manually started.

main

Output artifact format

Choose the output artifact format.

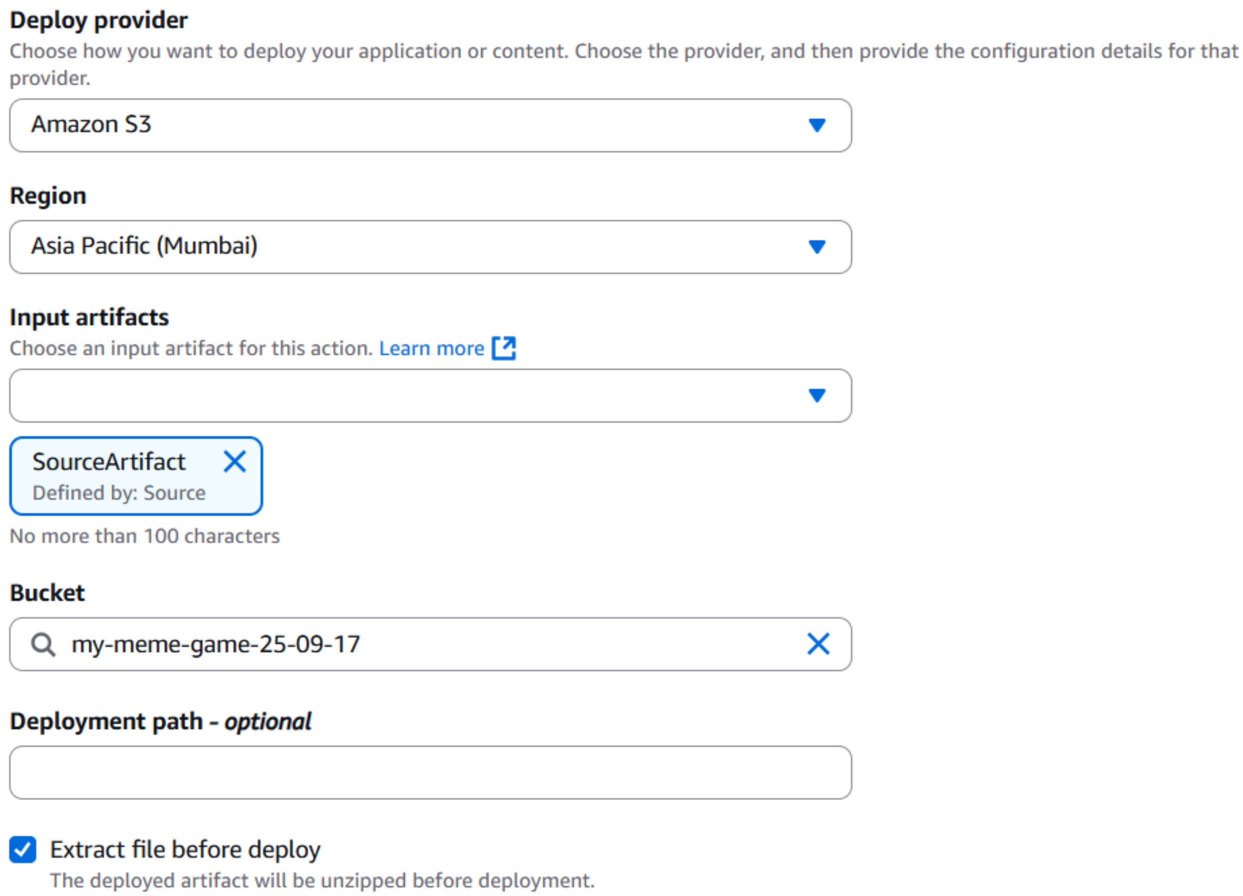
● CodePipeline default
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.

○ Full clone
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions. Learn more

Source stage configurations

Skip the Build Stage: Since this is a static website, there's no need for a build stage. Confirm that you want to skip this step.

Configure the Deploy Stage: Select Amazon S3 as the deploy provider. Choose the AWS region and the S3 bucket you created. Check the box for "Extract file before deploy" to ensure the files are correctly placed in the bucket.



Deploy provider

Choose how you want to deploy your application or content. Choose the provider, and then provide the configuration details for that provider.

Amazon S3 ▼

Region

Asia Pacific (Mumbai) ▼

Input artifacts

Choose an input artifact for this action. Learn more

▼

SourceArtifact ✕
Defined by: Source

No more than 100 characters

Bucket

my-meme-game-25-09-17 ✕

Deployment path - ***optional***

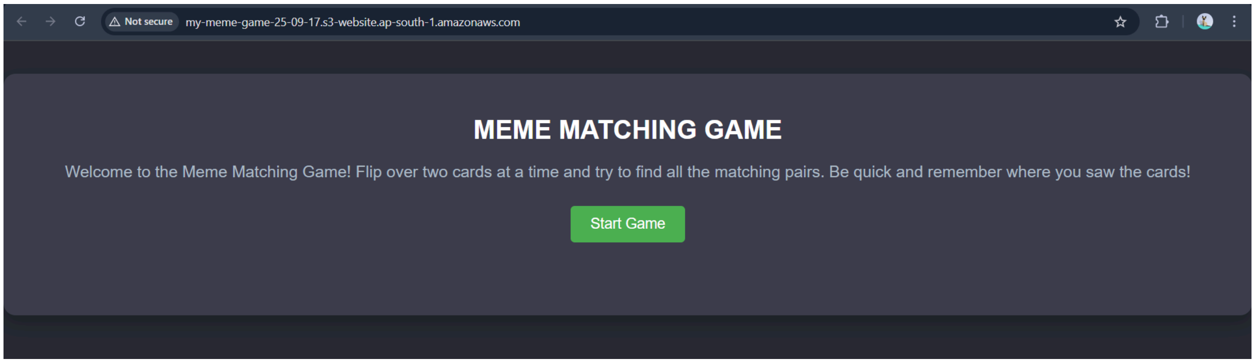
☑ Extract file before deploy
The deployed artifact will be unzipped before deployment.

Deploy stage configuration

Review and Create: Review all the settings and create the pipeline. The pipeline will automatically start and deploy the initial code.

Step 3.4: Testing and Validation

Access the Deployed Website: Go back to the S3 bucket's "Properties" tab and scroll down to "Static website hosting." Click the provided endpoint URL to see your game live on the web.



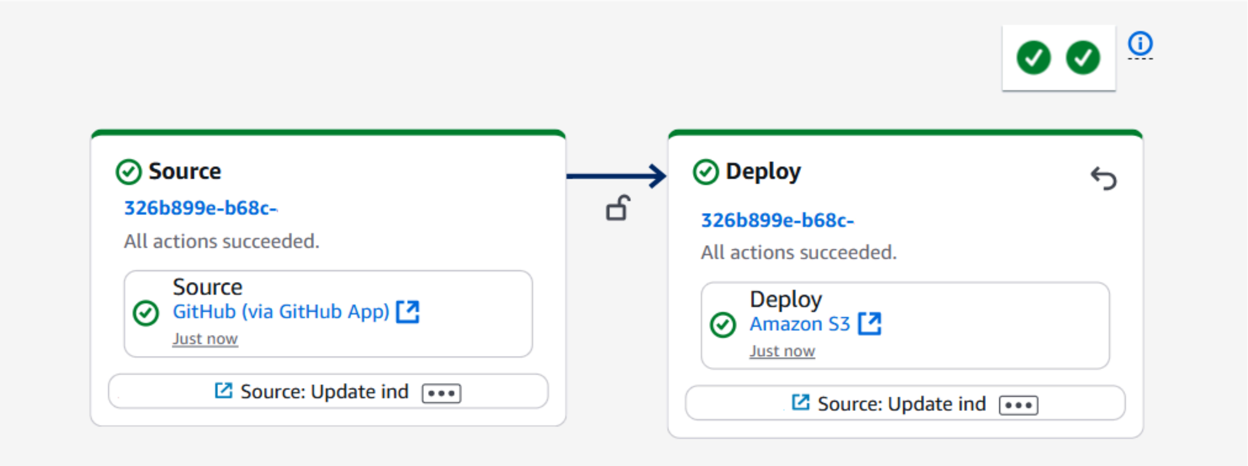
Not secure my-meme-game-25-09-17.s3-website.ap-south-1.amazonaws.com

MEME MATCHING GAME

Welcome to the Meme Matching Game! Flip over two cards at a time and try to find all the matching pairs. Be quick and remember where you saw the cards!

Final deployed game

Test Continuous Deployment: Make a small change to the index.html file directly in your GitHub repository and commit the change. The pipeline should automatically detect this change and redeploy the updated code to the S3 bucket. Refresh your website to see the update.



Source
326b899e-b68c-
All actions succeeded.

Source
GitHub (via GitHub App)
Just now

Source: Update ind

Deploy
326b899e-b68c-
All actions succeeded.

Deploy
Amazon S3
Just now

Source: Update ind

CodePipeline

4. Challenges and Solutions

Challenge: Initial Access Denied errors when trying to view the website.
Solution: This is a common issue related to incorrect permissions. The fix involves ensuring the S3 bucket policy has the necessary permissions to allow public read access (s3:GetObject).

5. Conclusion

This project successfully demonstrates the power of a continuous deployment pipeline using AWS services. By automating the deployment process, updates to the game can be pushed live almost instantly with a simple code commit to GitHub, saving time and effort. This showcases fundamental DevOps practices.

6. Future Improvements

Implement a build stage using AWS CodeBuild to perform tasks like minifying code or running unit tests before deployment.
Extend the pipeline to deploy to a more scalable service like Elastic Beanstalk or an EC2 instance.