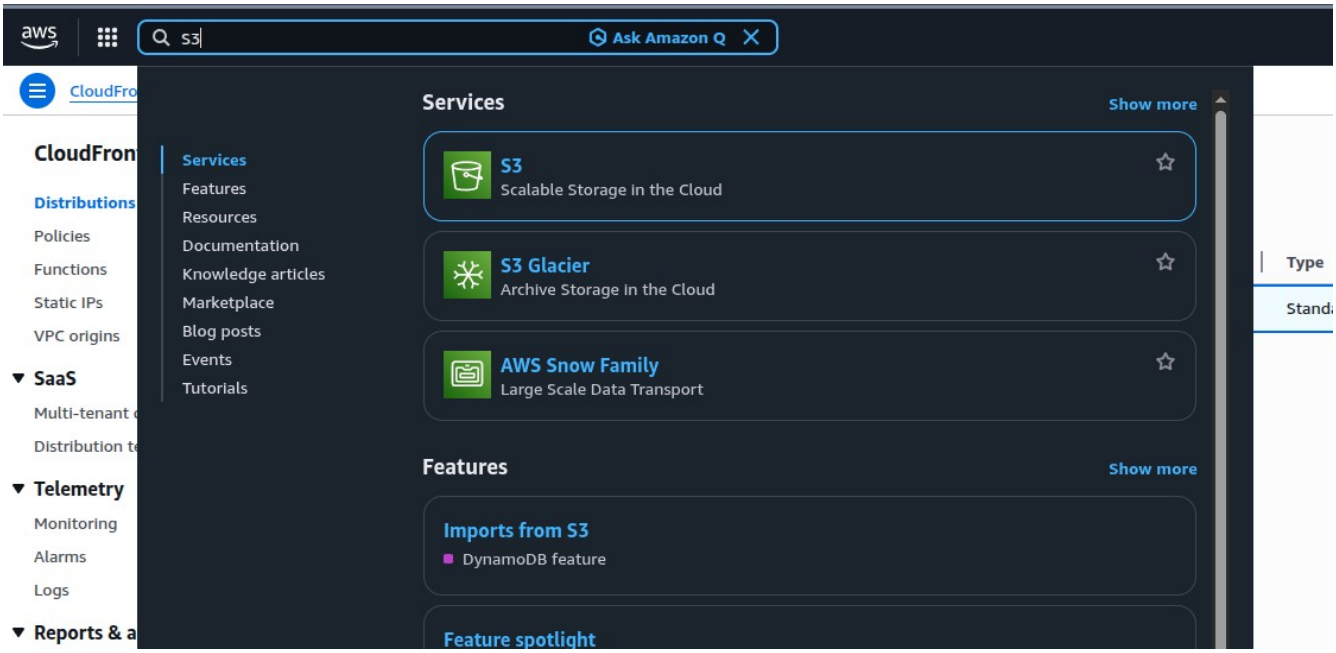
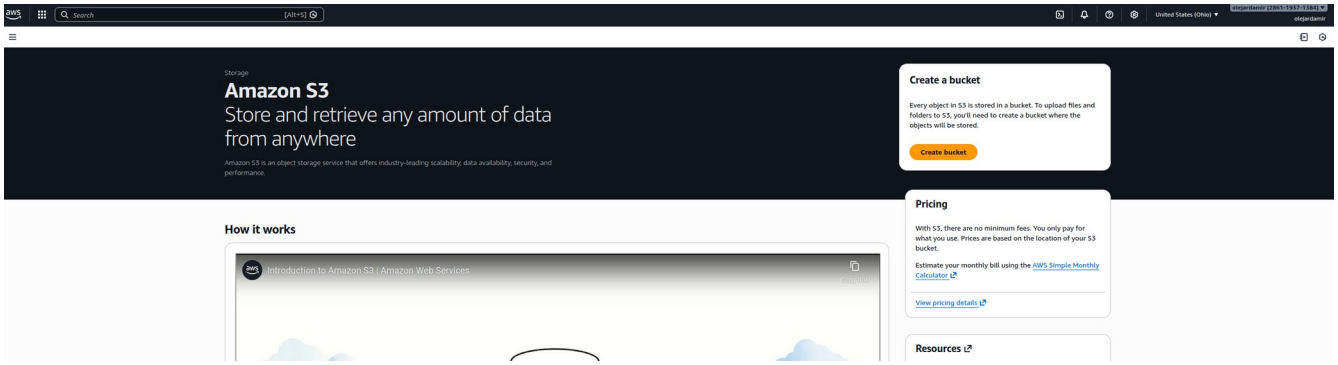
STEP 1
First, open an account with AWS, they may offer $200 free for first-time users.

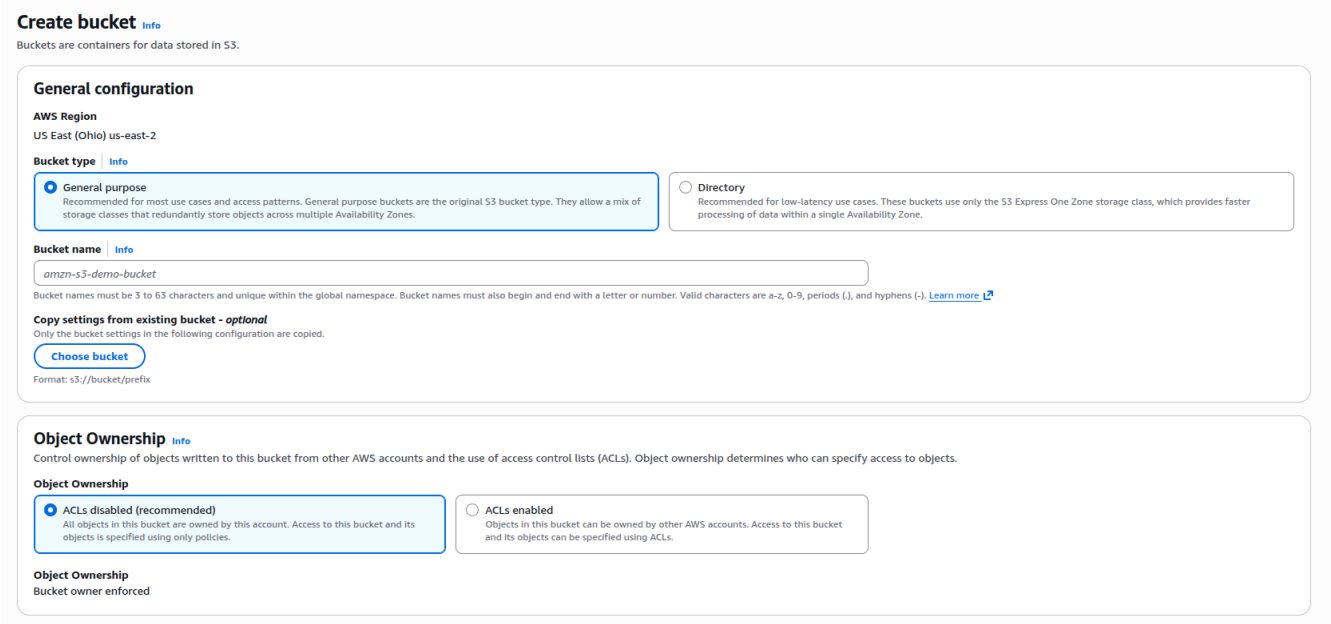
STEP 2
Now, we need to create an S3 bucket.
In the upper left corner where the search is found, type S3 and click on the blue letters that say S3.



Now, assuming this is the first-time use, click on the orange “Create Bucket” button placed in a counter-intuitive position:



You will see a Generate Bucket page that looks like this:



Follow these steps:

1. AWS Region

- **Leave as is, for example**:
  US East (Ohio) – us-east-2

This is fine. Just remember it. You will reuse it later.

---

2. Bucket type

- Select: **General purpose**

Do **not** choose Directory.

---

3. Bucket name

Enter a globally unique name, for example:

my-sudoku-web

If AWS complains, slightly change the name (e.g. add -2026).

---

4. Copy settings from existing bucket

- **Leave empty**
- Do **not** select anything

---

5. Object Ownership

Select:

- **ACLs disabled (recommended)**
- **Bucket owner enforced**

This is **mandatory**.
This prevents permission problems during GitHub uploads.

---

6. Block Public Access (very important)

- Ensure **Block all public access** is **ON**
- All four checkboxes must be **checked**

Do **not** uncheck anything here.

This bucket must be **private**.

---

7. Bucket Versioning

- Select: **Disable**

You do **not** need versioning for a static site.

---

8. Tags

- Leave empty
- Optional, skip for now

---

9. Default encryption

Select:

- **Server-side encryption with Amazon S3 managed keys (SSE-S3)**

Set **Bucket Key** as **Disable**.

This is correct and cost-safe.

---

10. Advanced settings

- Do **nothing**
- Do **not** enable static website hosting here

We will **not** use S3 website hosting at all.

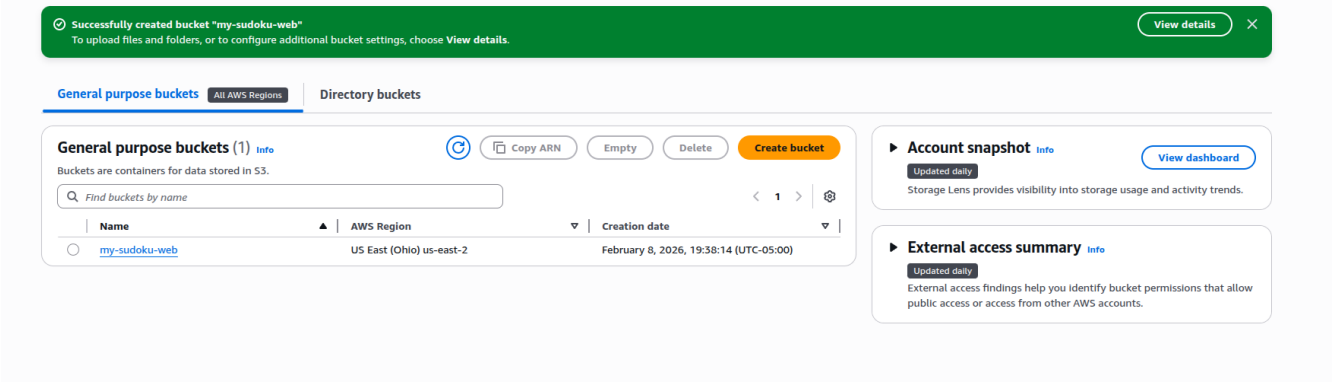
---

11. Create the bucket

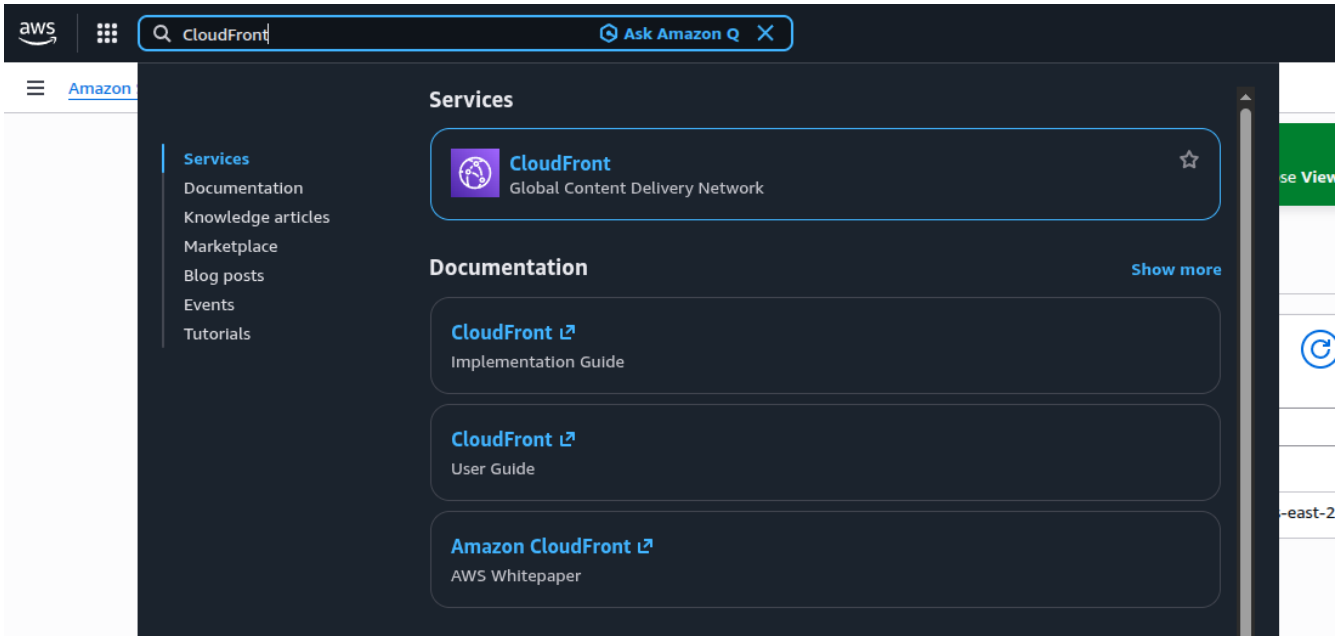
Click:

- **Create bucket**

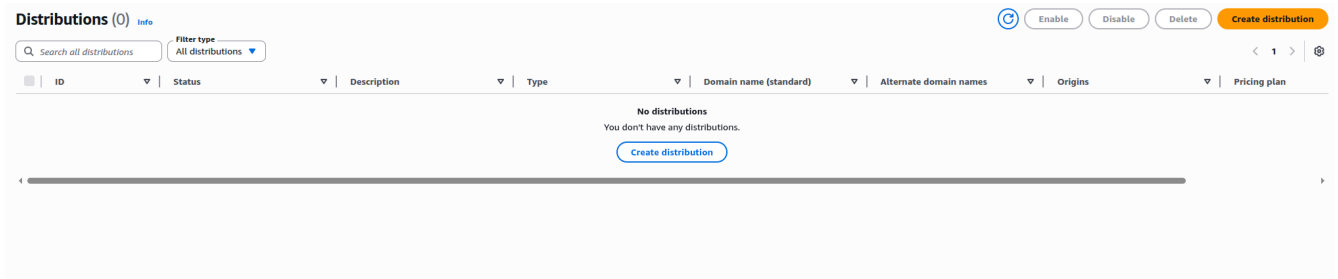
Now, as a confirmation, you will see something like this:



Now, we need to setup CloudFront, so we do the same thing, we search for it...

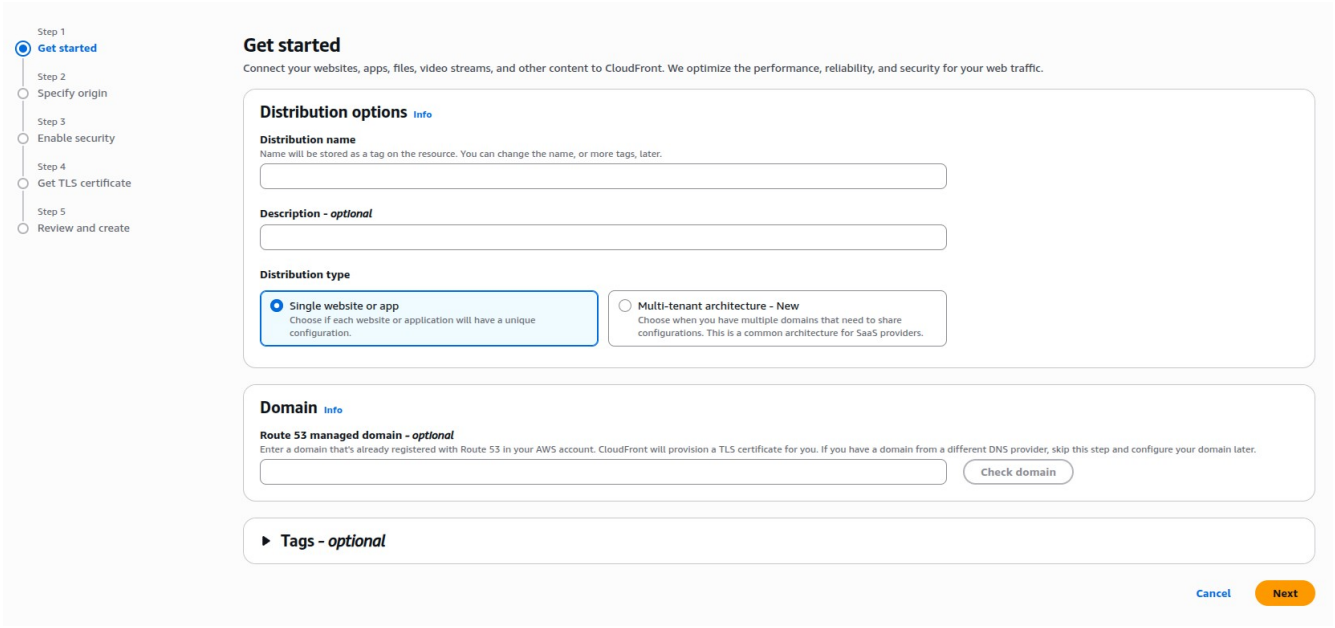


Click on the blue letters, and you will see the page that looks like this:



Click on the Create Distribution, close popups, if any.

So, lets start creating a distribution. You will see something like:



1. Distribution name

- Enter something simple and descriptive, for example:

sudoku-web

This is just a tag. It does not affect URLs.

---

2. Description (optional)

- Leave empty or add:

Sudoku Vite app (S3 + CloudFront)

Optional. No functional impact.

---

3. Distribution type

- Keep **Single website or app**
  Do **not** select Multi-tenant.

---

4. Domain (Route 53 managed domain)

- **Leave this empty**
- Do **not** enter any domain

- Do **not** click “Check domain”

We will use the default CloudFront domain for now. Custom domains come later.

---

5. Tags

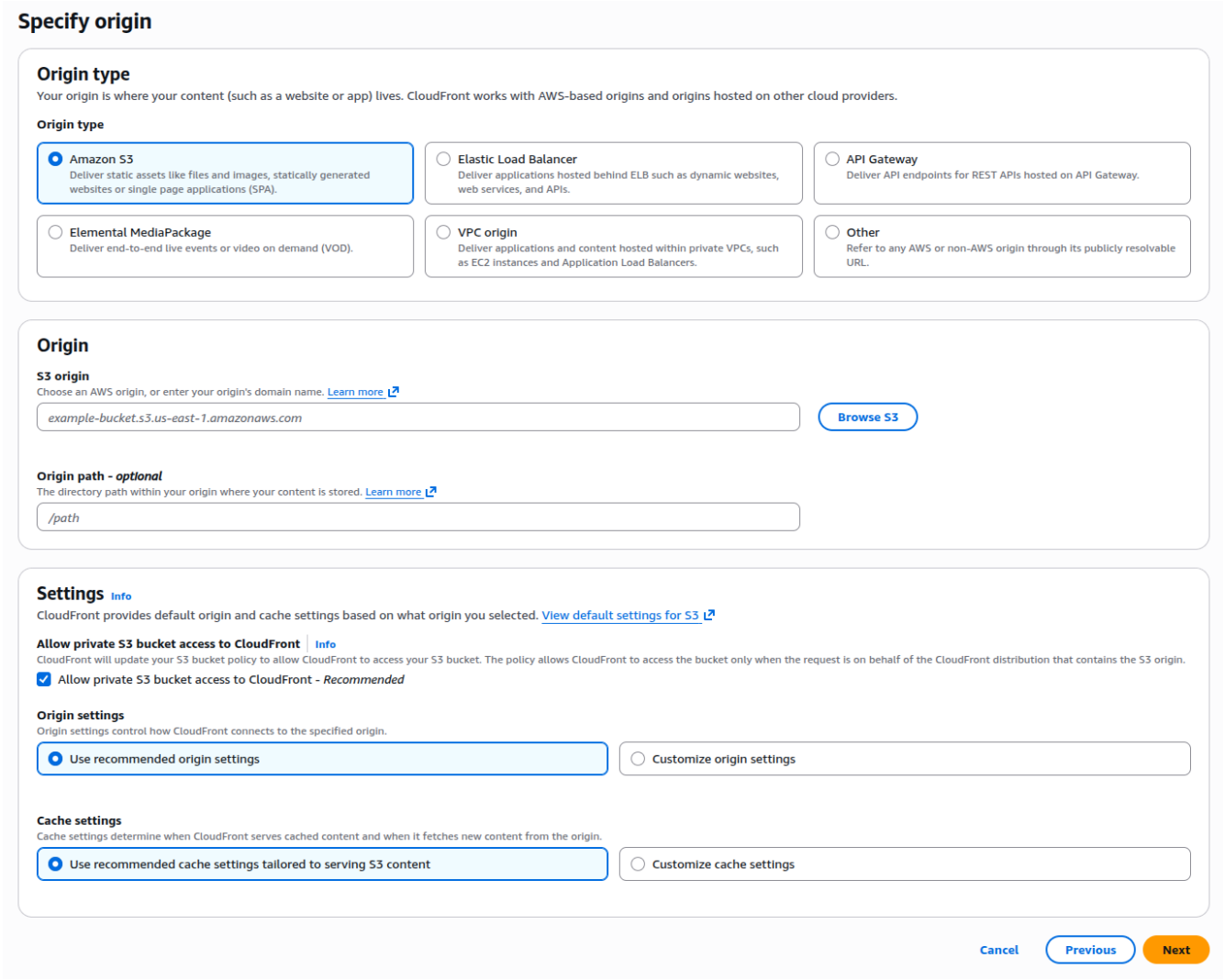
- Leave everything empty
- Skip

---

6. Proceed

Click **Next** (bottom right).

Now, you will need to specify the origin in a view like this one:



1. Origin Type is S3, allow private S3 bucket access to CloudFront

You should see this **checked**:

Allow private S3 bucket access to CloudFront (Recommended)

Leave this ON. Do not change it.

What this does:

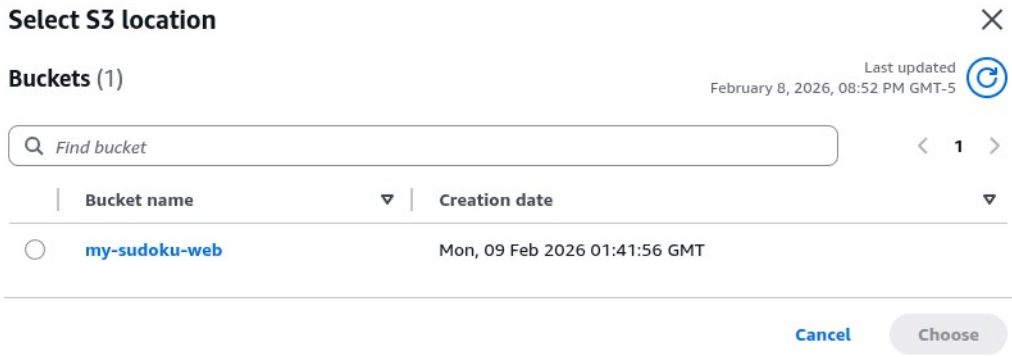
- CloudFront creates an **Origin Access Control (OAC)**
- CloudFront will later **update your S3 bucket policy automatically**
- Your bucket stays private
- No public S3 access is ever enabled

This is exactly what we want.

---

2. Origin settings

Click on Browse S3 and choose your sudoku origin



---

3. Cache settings

Selected option should be:

Use recommended cache settings tailored to serving S3 content

Leave it as-is.

Why:

- Static assets (.js, .css, .png) are cached efficiently
- index.html still respects invalidations
- This works perfectly with SPA + CloudFront invalidation

We will **not** customize TTLs yet.

---

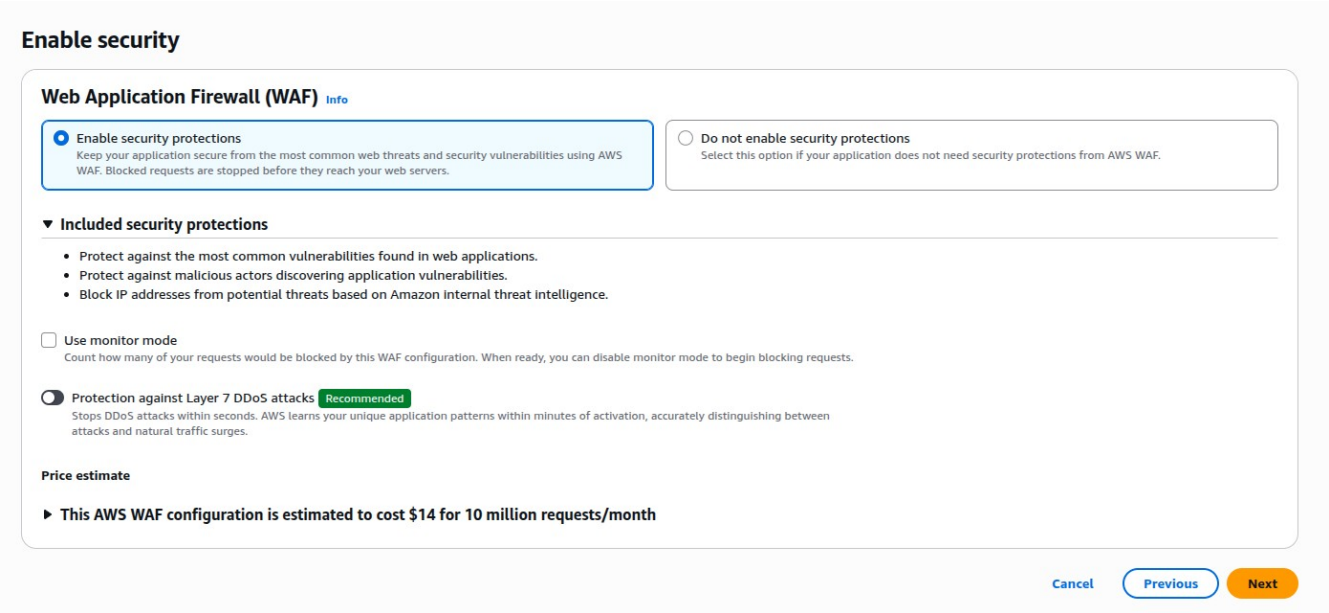
4. Do NOT customize anything here

---

5. Proceed

Click **Next** (bottom right).

Now, we will need to enable the security as the next step:



1. Web Application Firewall (WAF)

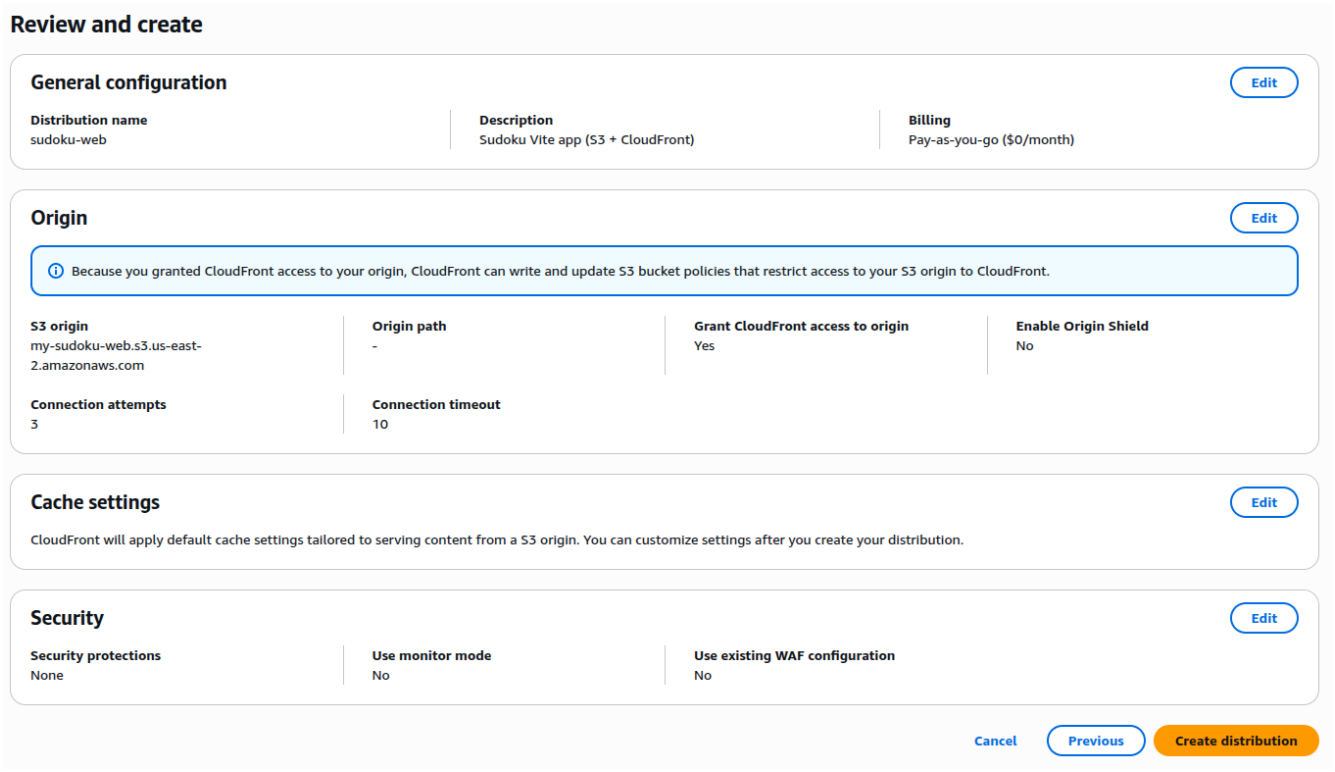
Select:

Do not enable security protections

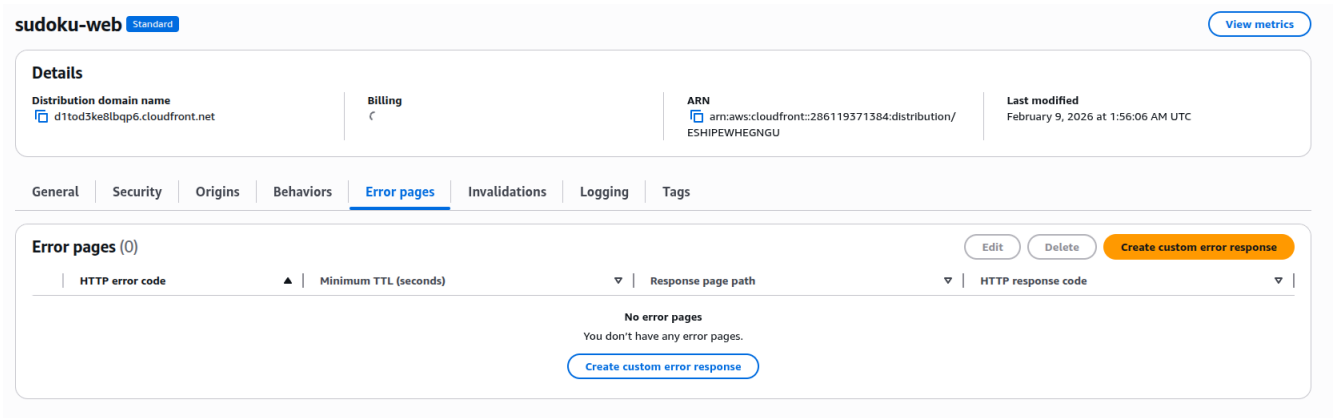
Why:

- AWS WAF **costs money** (~$14/month shown)
- Click **Next**.

Finally, you should be able to create the distribution, click on the Create distribution button:



Next, you will see a screen like this one:



1. Click the **Error pages** tab
2. Click **Create custom error response**

First error

- **HTTP error code**: 403
- **Customize error response**: Yes

- **Response page path**: /index.html
- **HTTP response code**: 200: OK

Click **Create**.

Second error

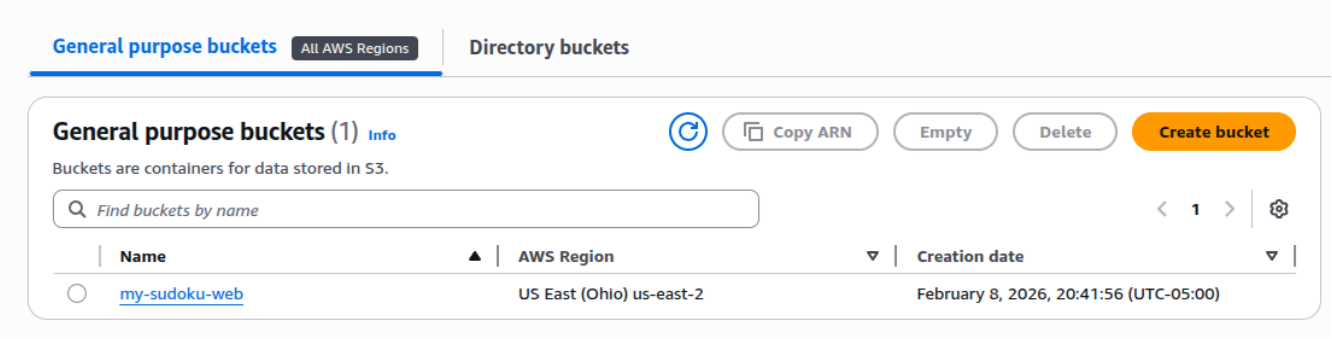
Repeat **Create custom error response** with:

- **HTTP error code**: 404
- **Customize error response**: Yes
- **Response page path**: /index.html
- **HTTP response code**: 200: OK

Click **Create**.

- CloudFront will now:
    - Serve index.html at /
    - Serve index.html for all client-side routes
- This is **required** for React / Vite
- These changes may take **a few minutes**

Now, we need to go **back to our S3 bucket**, you can do the search and select S3 like before.



Now, click on the my-sudoku-web bucket (blue underlined letters).

Now, this is extremely important! We will need to build a project and if we do not setup vite.config.ts, our page will not run!

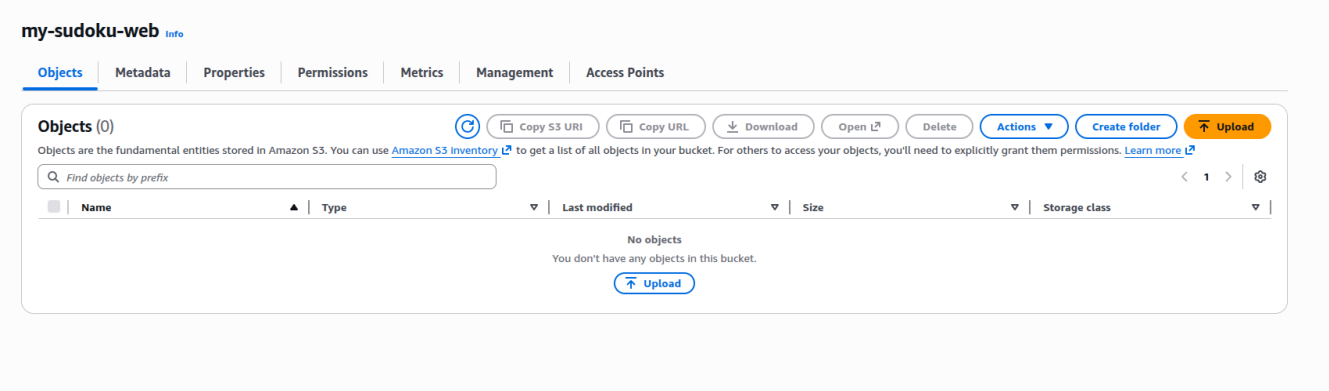
Make sure that you got this file:
sudoku/web/vite.config.ts

With exactly this:

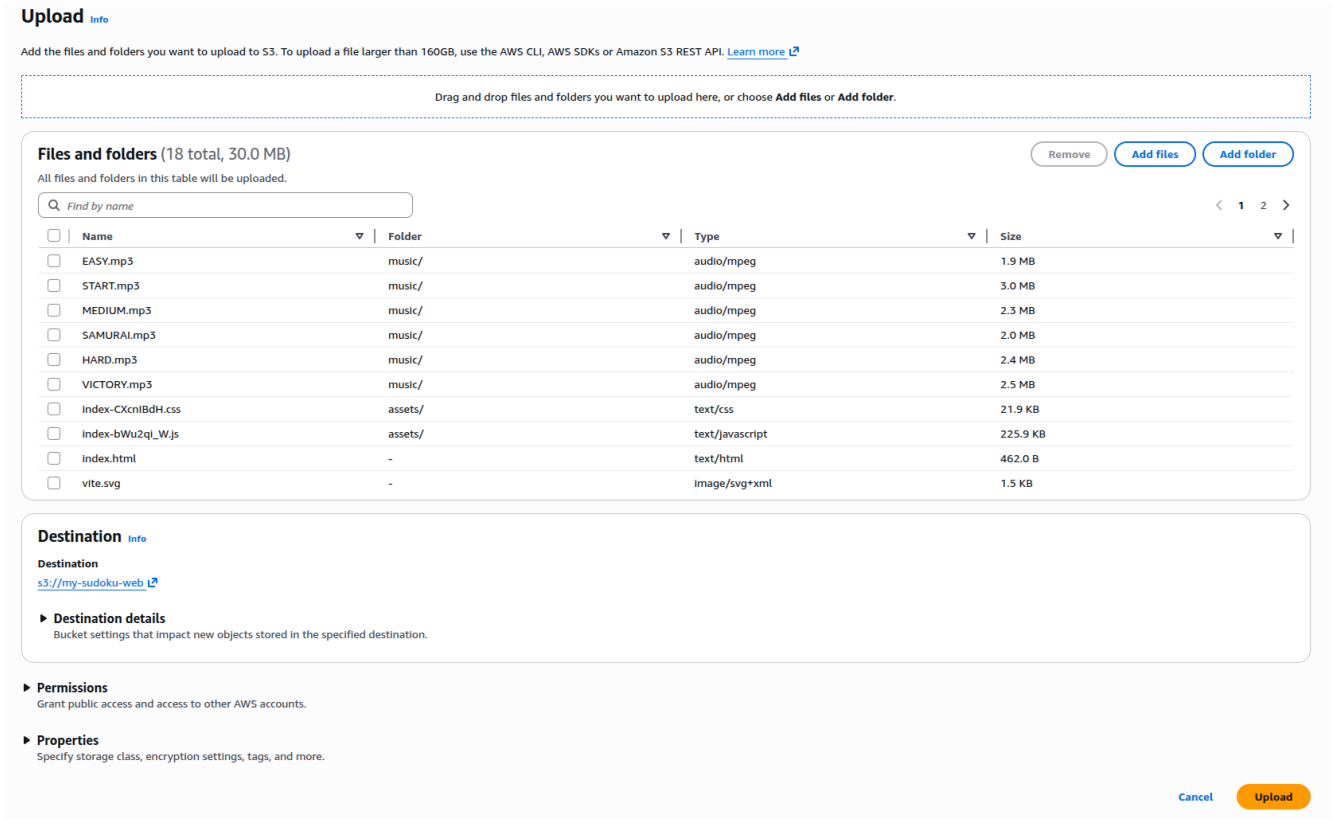
```
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react";

export default defineConfig({
  plugins: [react()],
  base: "/",
});
```

Then from sudoku/web/ do:
npm build

We have all we need to upload, so lets upload it. After creating a bucket, you will see the following... click on the Upload button:

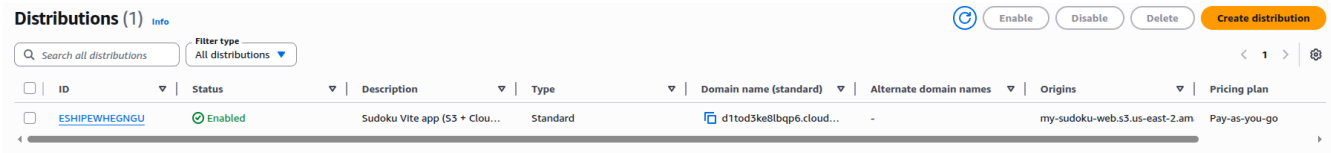


Now, here is a tricky part where AWS should have made it more flexible and functional. You will need to upload files AND folders. So, first click on **Add files** button, go to your **dist** folder which has been generated by npm build command (for example: .../sudoku/web/dist) and select all files (not folders) and press OK. Then, add folders assets, and music folders separately by pressing **Add folder**, for each folder. It should look like this:



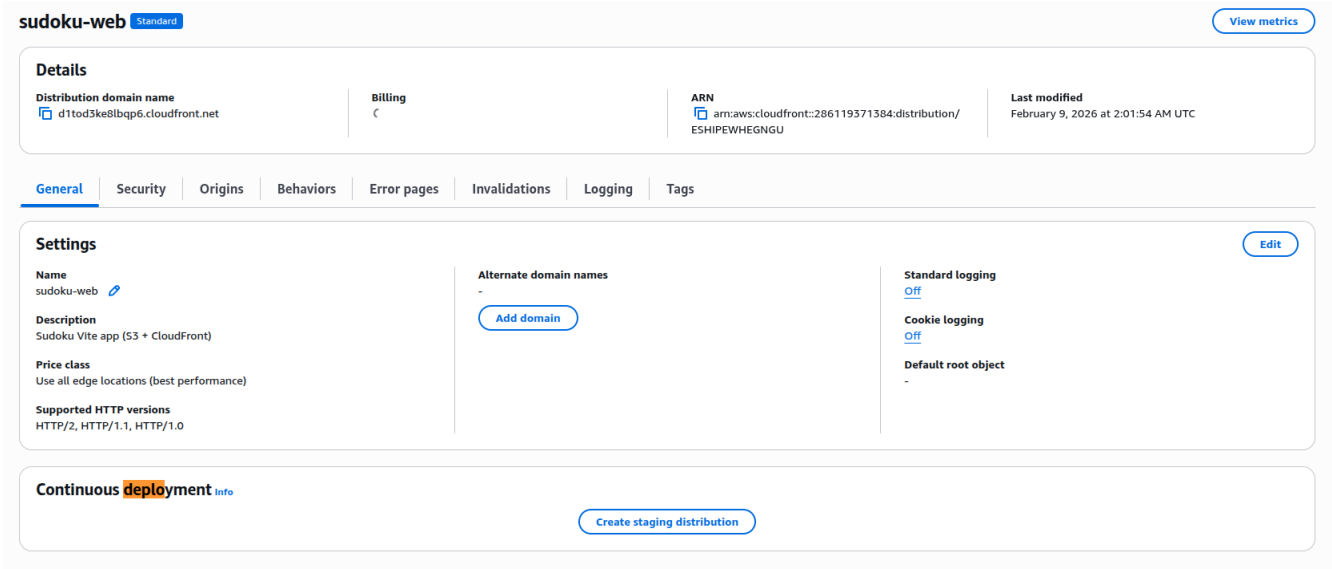
Press **Upload** button.
Wait for the upload to complete, and press the orange “Close” button when it does.

Go back to Cloudfront to see your distributions:



Click on the one which was meant for the game.

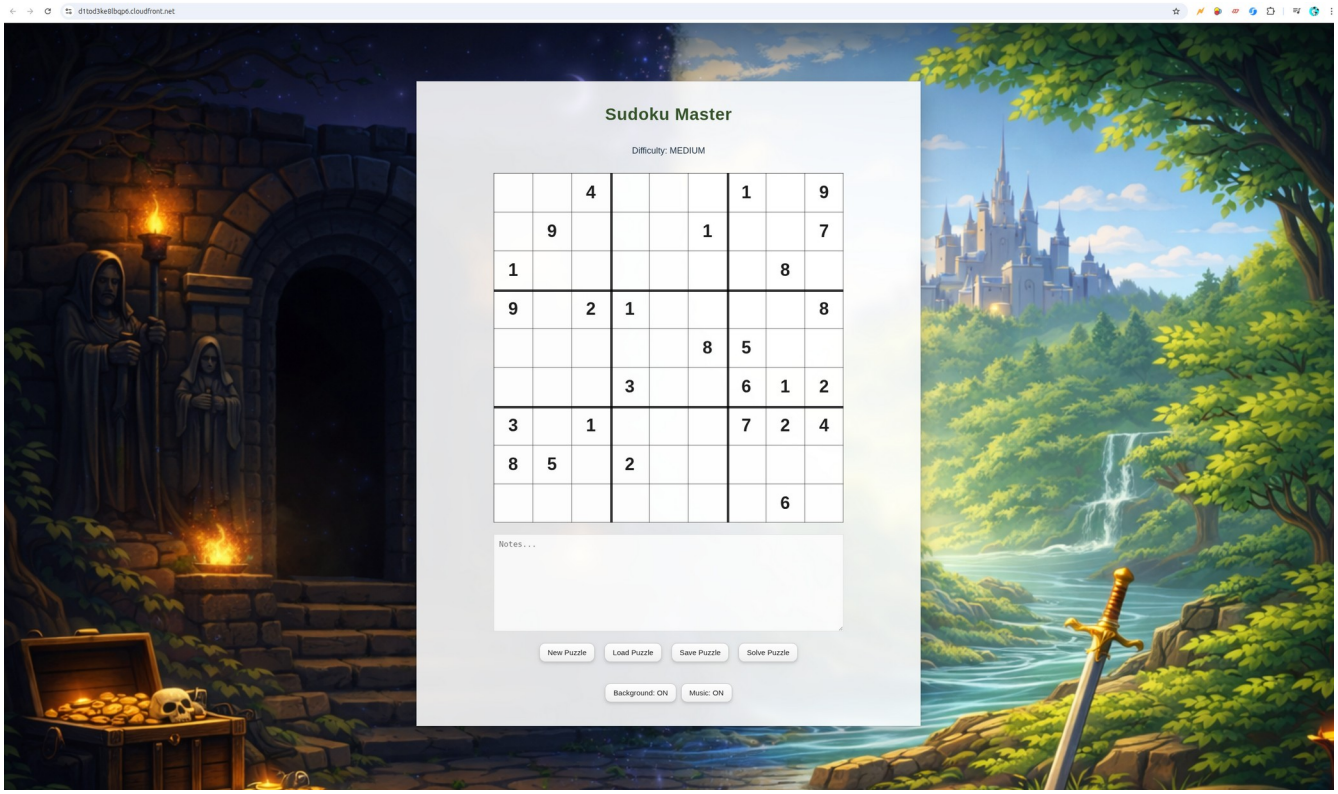
Finally, you will see this:



Copy the distribution name, in my case "d1tod3ke8lbqp6.cloudfront.net" and go to

https://d1tod3ke8lbqp6.cloudfront.net

The game is ON, HAPPY GAMING !

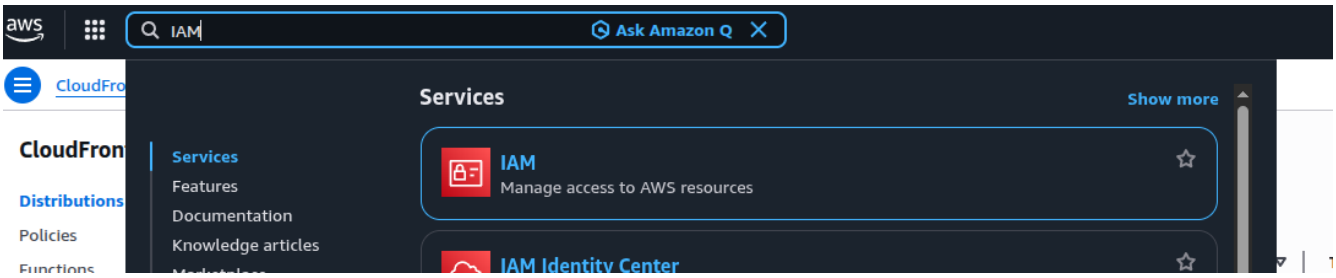


Please note, I have removed this so I do not get charged. I have done another pipeline deployment on github.io, which was free, see: https://olejardamir.github.io/sudoku/

Now that **manual deployment works**, GitHub Actions becomes easy. We want to do this entablement so that each time we make a change to a code we can do automated deployment.

1. IAM role for GitHub Actions (OIDC)

We use the same trick and open the IAM on AWS:



2. Create the OIDC identity provider (one-time)

You only do this **once per AWS account**.

Go here

1. In IAM, click **Identity providers** (left sidebar)
2. Click **Add provider** (orange button on far right)

Fill the form

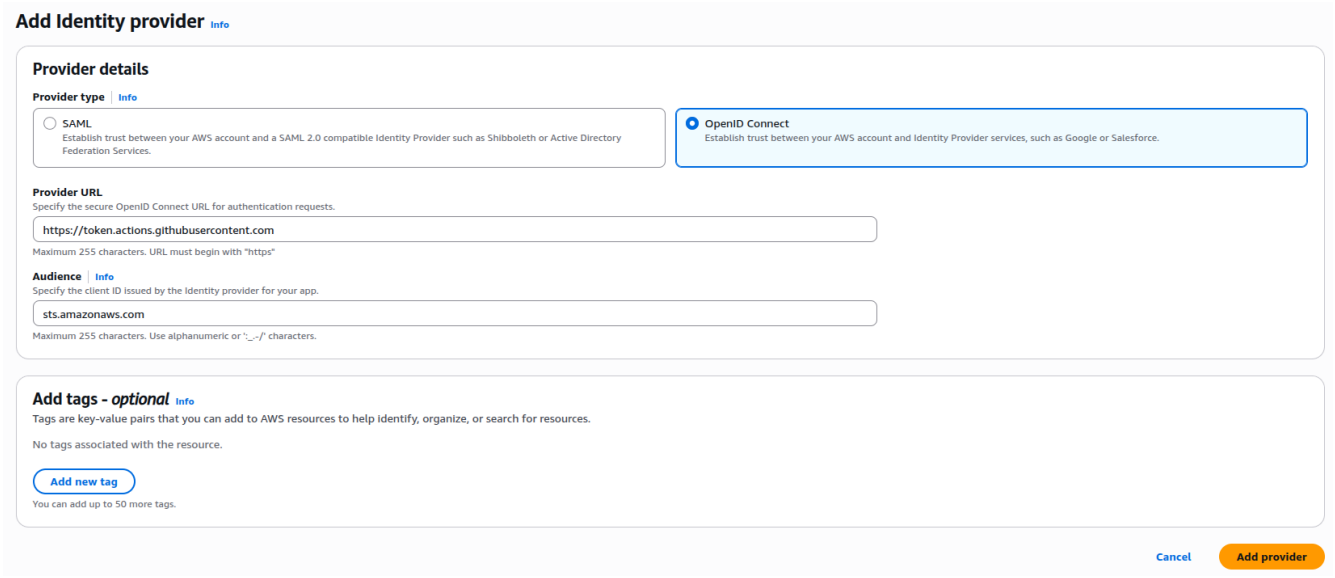
- **Provider type**: OpenID Connect
- **Provider URL**:

  https://token.actions.githubusercontent.com

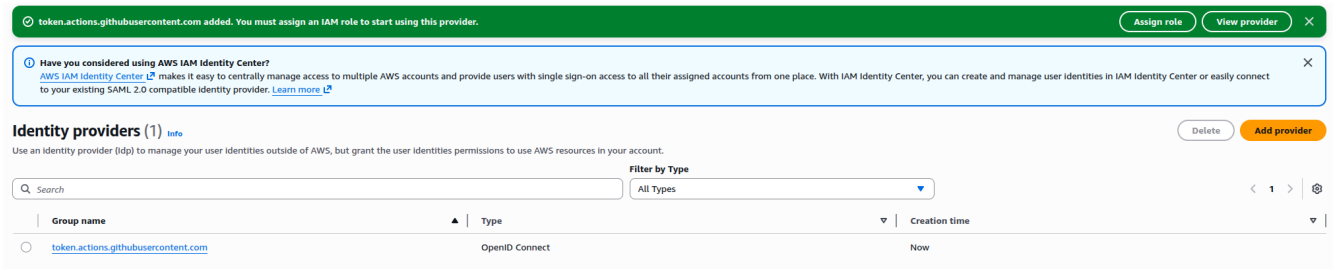
- **Audience**:

  sts.amazonaws.com

Click **Add provider**.



You should now see token.actions.githubusercontent.com listed.



---

3. Create the IAM role (this is the important part)

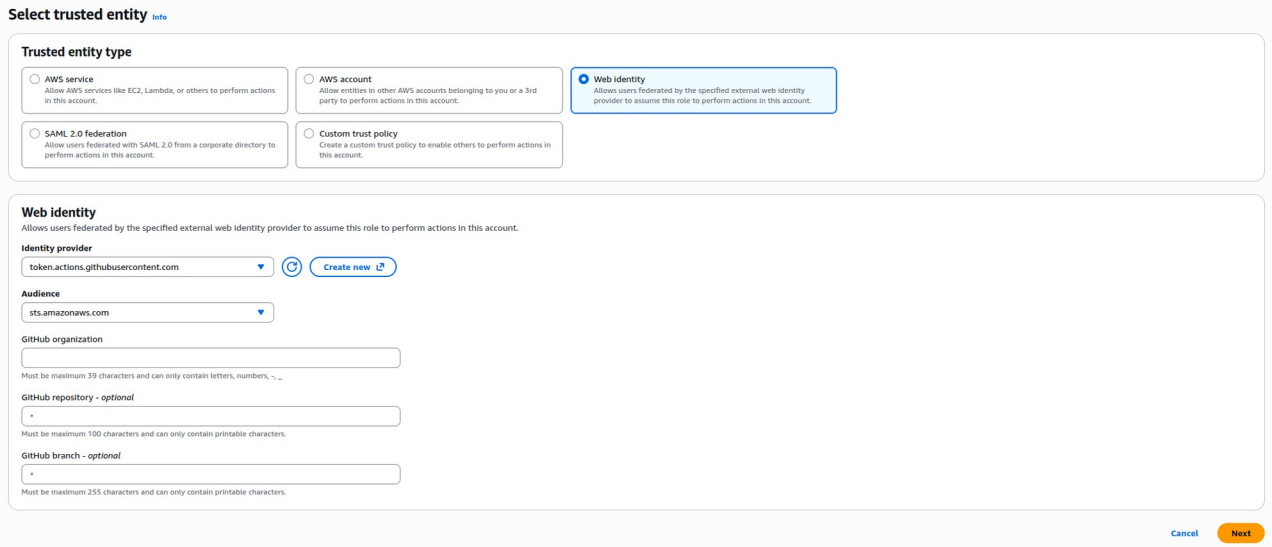
Go here

1. IAM → **Roles (on the left sidebar)**
2. Click **Create role (orange button)**

Step 3.1 — Trusted entity

- Select **Web identity**

- Identity provider:
  token.actions.githubusercontent.com
- Audience:
  sts.amazonaws.com
- Organization:
  YOUR GITHUB USERNAME



Click **Next**.

Step 3.2 — Permissions

Click **Next** *without selecting anything*.

We will attach a custom policy after the role exists.

Step 3.3 — Role name

- **Role name** (example):

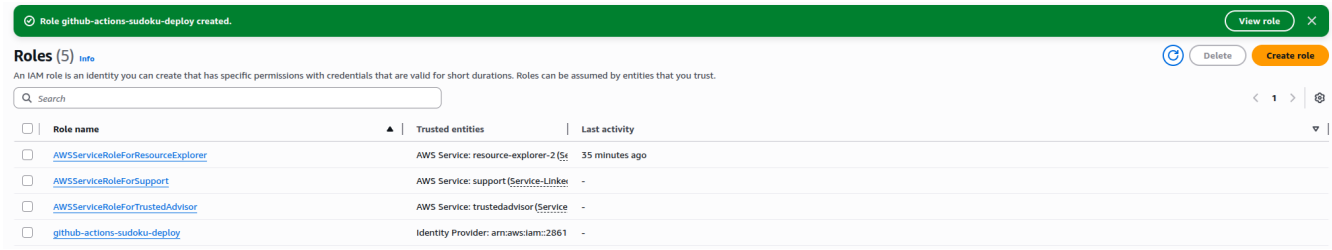
  github-actions-sudoku-deploy

Click **Create role**.

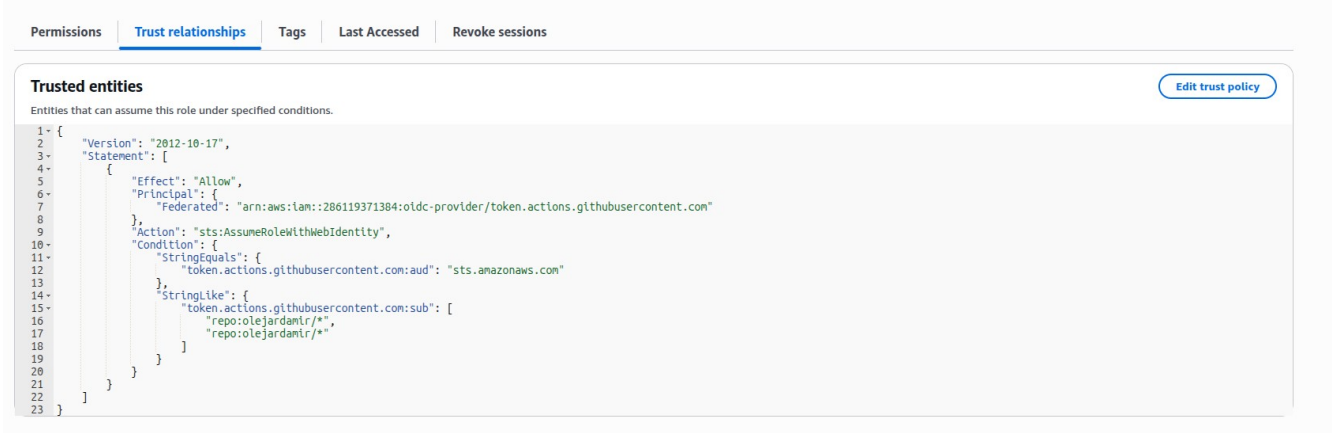
---

4. Edit the trust policy (critical, do not skip)

IAM → Roles → click your role



Go to **Trust relationships** → Edit trust policy



Replace <ACCOUNT_ID>, <ORG_OR_USER>, <REPO>

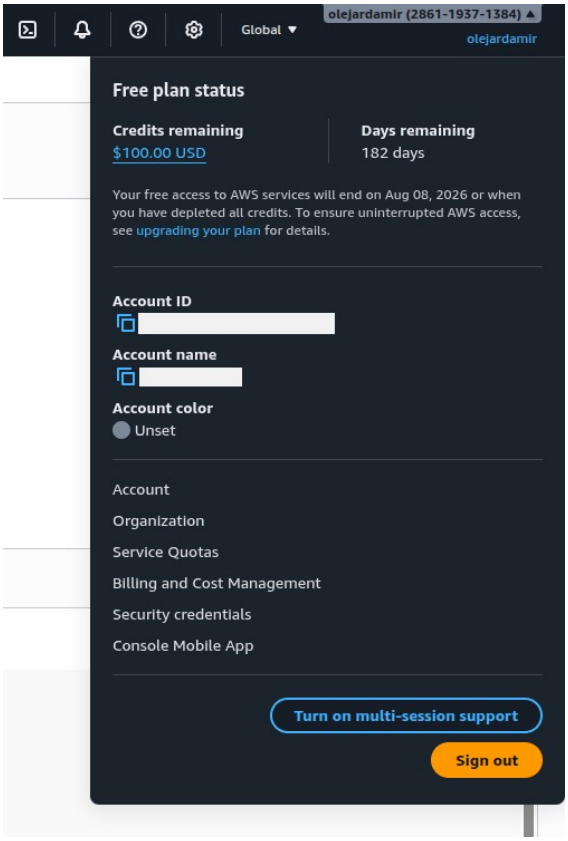
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated":
"arn:aws:iam::<ACCOUNT_ID>:oidc-provider/token.actions.githubusercontent.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
        },
        "StringLike": {
          "token.actions.githubusercontent.com:sub":
"repo:<ORG_OR_USER>/<REPO>:ref:refs/heads/main"
```

```
      }
    }
  }
 ]
}
```

1. The <ACCOUNT_ID>

This is your **12-digit AWS Account Number**.

- **Where to find it:** Look at the top right corner of your **AWS Management Console**. It's usually displayed next to your username/alias.
- **Format:** It should be a plain string of numbers (e.g., 123456789012).



2. The <ORG_OR_USER><REPO>

This is your **GitHub Organization name** or your **GitHub Username**.

- **Where to find it:** Go to your repository on GitHub. Look at the URL: github.com/ORG_OR_USER/REPO.
- **Example:** If your repo is https://github.com/olejardamir/sudoku, your <ORG_OR_USER> is olejardamir and <REPO> is sudoku.

Click **Update policy**.

---

5. Attach the permissions policy (S3 + CloudFront)

Create the policy

1. IAM → **Policies (right hand side panel)**
2. Click **Create policy (orange button)**
3. Switch to **JSON** tab, MAKE SURE TO REPLACE <ACCOUNT_ID>:distribution/<DISTRIBUTION_ID> with actual IDs.

Paste this (replace bucket + IDs):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": "arn:aws:s3:::my-sudoku-web"
    },
    {
      "Effect": "Allow",
      "Action": ["s3:PutObject", "s3:DeleteObject", "s3:GetObject"],
      "Resource": "arn:aws:s3:::my-sudoku-web/*"
    },
    {
      "Effect": "Allow",
      "Action": ["cloudfront:CreateInvalidation"],
      "Resource": "arn:aws:cloudfront::<ACCOUNT_ID>:distribution/<DISTRIBUTION_ID>"
    }
  ]
}
```

Click **Next → Next**

- **Policy name**:

  github-actions-sudoku-deploy-policy

Click **Create policy**.

Attach policy to the role

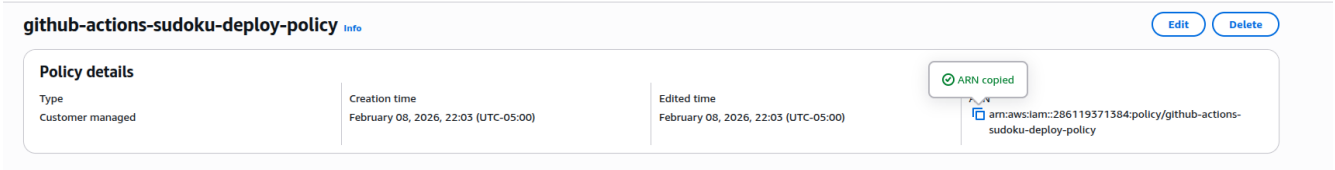
1. IAM → **Roles**

2. Click github-actions-sudoku-deploy
3. **Permissions** tab
4. Click **Add permissions → Attach policies**
5. Select (use search, there are too many policies):
    - github-actions-sudoku-deploy-policy
6. Click **Add permissions**

---

6. Copy the Role ARN (you will need this)

On the role summary page, copy:



arn:aws:iam::<ACCOUNT_ID>:role/github-actions-sudoku-deploy

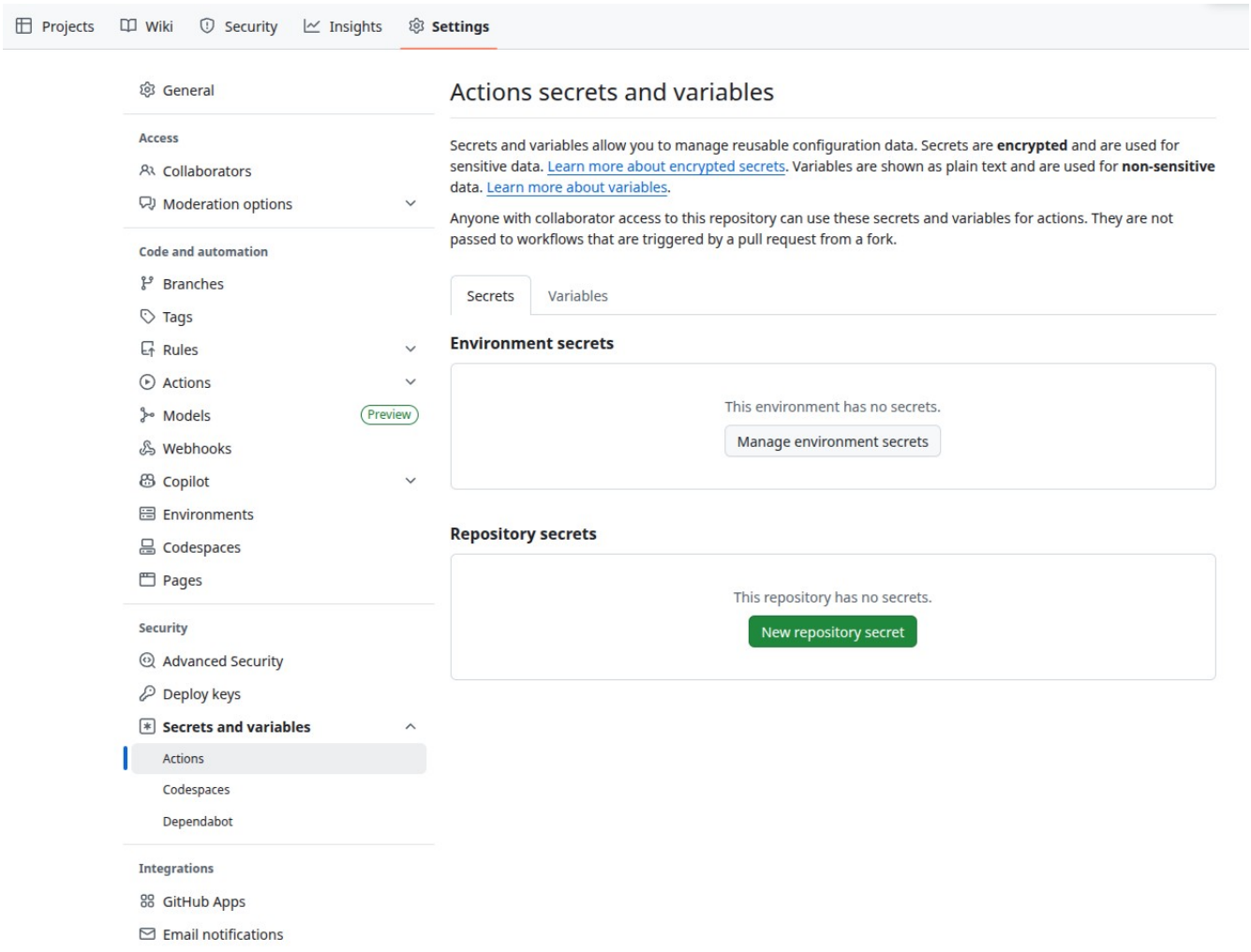
This goes into GitHub Secrets as AWS_ROLE_ARN.

This means that we are done with AWS and can move onto **Github**.

Add GitHub Repository Secrets

Go to:

GitHub → Your repo → Settings → Secrets and variables → Actions



Click **New repository secret**

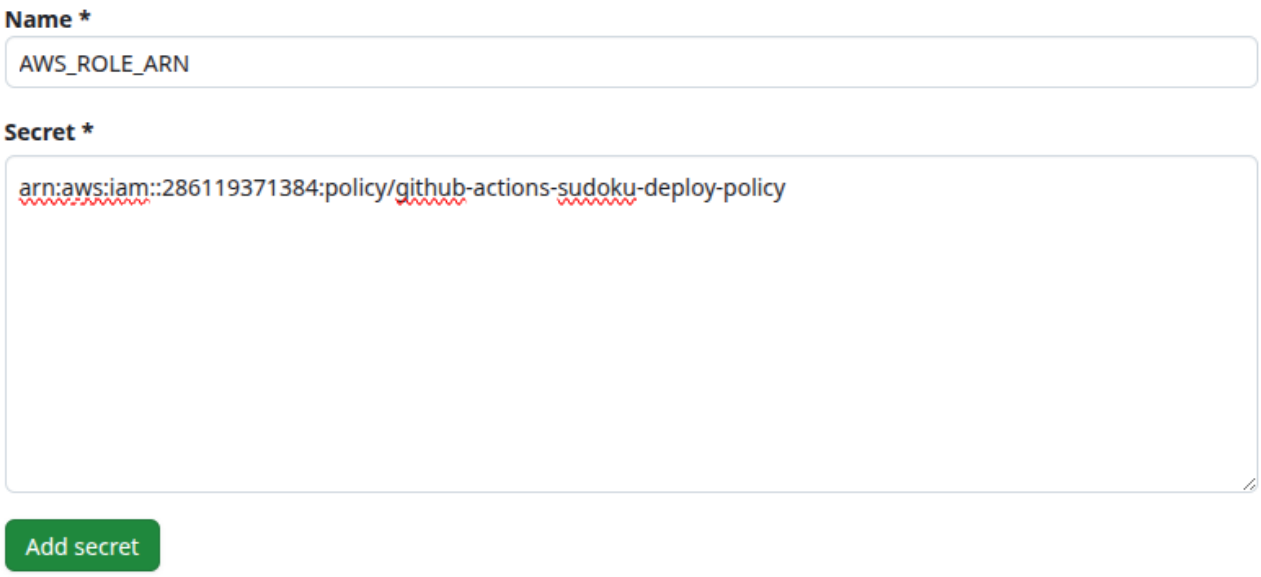
Create the following secrets:

1. AWS_ROLE_ARN

Value = the full role ARN you copied

In this case:

arn:aws:iam::286119371384:role/github-actions-sudoku-role



2. AWS_REGION

Value: us-east-2

(Or whatever your bucket region was in AWS, if needed, go back to AWS to verify)

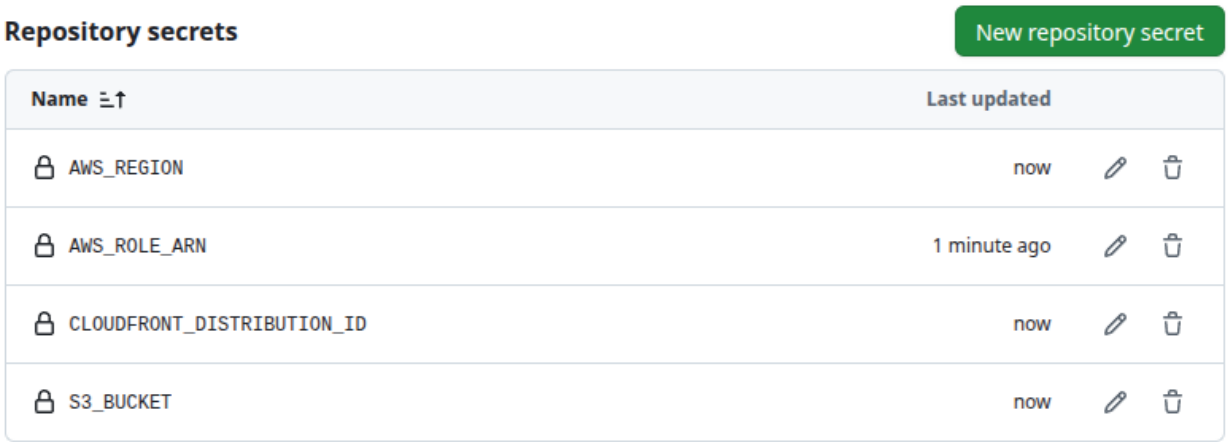
3. S3_BUCKET

Value: my-sudoku-web

4. CLOUDFRONT_DISTRIBUTION_ID

Value: E34B6UAED3H6MI

When finished, you should see 4 secrets in the list.



Now, you will need to go back to the original code. You will need to locate the file .github/workflows/deploy.yml or create directories first, and then create a new file if it does not exist.

Add this as a file content, MAKE SURE TO USE PROPER IDs, otherwise it will not work :

```
name: Deploy to AWS S3 + CloudFront


on:
  push:
    branches:
      - main
permissions:
  id-token: write
  contents: read
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repo
        uses: actions/checkout@v4


      - name: Setup Node
        uses: actions/setup-node@v4
        with:
          node-version: 20
          cache: npm
          cache-dependency-path: web/package-lock.json


      - name: Install dependencies
        working-directory: web
```

```
    run: npm ci

  - name: Build
    working-directory: web
    run: npm run build

  - name: Configure AWS credentials
    uses: aws-actions/configure-aws-credentials@v4
    with:
      role-to-assume: ${{ secrets.AWS_ROLE_ARN }}
      aws-region: ${{ secrets.AWS_REGION }}

  - name: Deploy to S3
    run: aws s3 sync web/dist s3://${{ secrets.S3_BUCKET }} --delete

  - name: Invalidate CloudFront
    run: aws cloudfront create-invalidation \
      --distribution-id ${{ secrets.CLOUDFRONT_DISTRIBUTION_ID }} \
      --paths "/*"
```

And then, you will need to push it to a repository. Please note, I have overridden this by doing another pipeline using github.io.

git add .

git commit -m "deploy"

git push origin main

CONGRATULATIONS! You are is now fully automated with AWS and Github CI/CD pipeline.