

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.

The screenshot shows the AWS search interface. In the search bar at the top, the text "s3" is typed. Below the search bar, the "Services" section is expanded, showing three items: "S3 Scalable Storage in the Cloud", "S3 Glacier Archive Storage in the Cloud", and "AWS Snow Family Large Scale Data Transport". To the right of the services, there is a vertical sidebar with sections for "Type" and "Standards". On the left side of the main content area, there is a sidebar with links for "CloudFront", "Distributions", "Policies", "Functions", "Static IPs", "VPC origins", "SaaS", "Telemetry", and "Reports & analysis".

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

The screenshot shows the Amazon S3 landing page. At the top, it says "Amazon S3" and "Store and retrieve any amount of data from anywhere". Below this, there is a brief description of what S3 is. On the right side, there is a large orange "Create a bucket" button. To the left of the button, there is a "How it works" section with a video thumbnail and some text. To the right of the button, there is a "Pricing" section with a note about no minimum fees and a link to the AWS Simple Monthly Calculator. At the bottom, there is a "Resources" section with a "View pricing details" link.

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

Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region
US East (Ohio) us-east-2

Bucket type Info

General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info
`amzn-s3-demo-bucket`

Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn more ↗](#)

Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.

Choose bucket

Format: s3://bucket/prefix

Object Ownership Info
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

Object Ownership

ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership
Bucket owner enforced

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

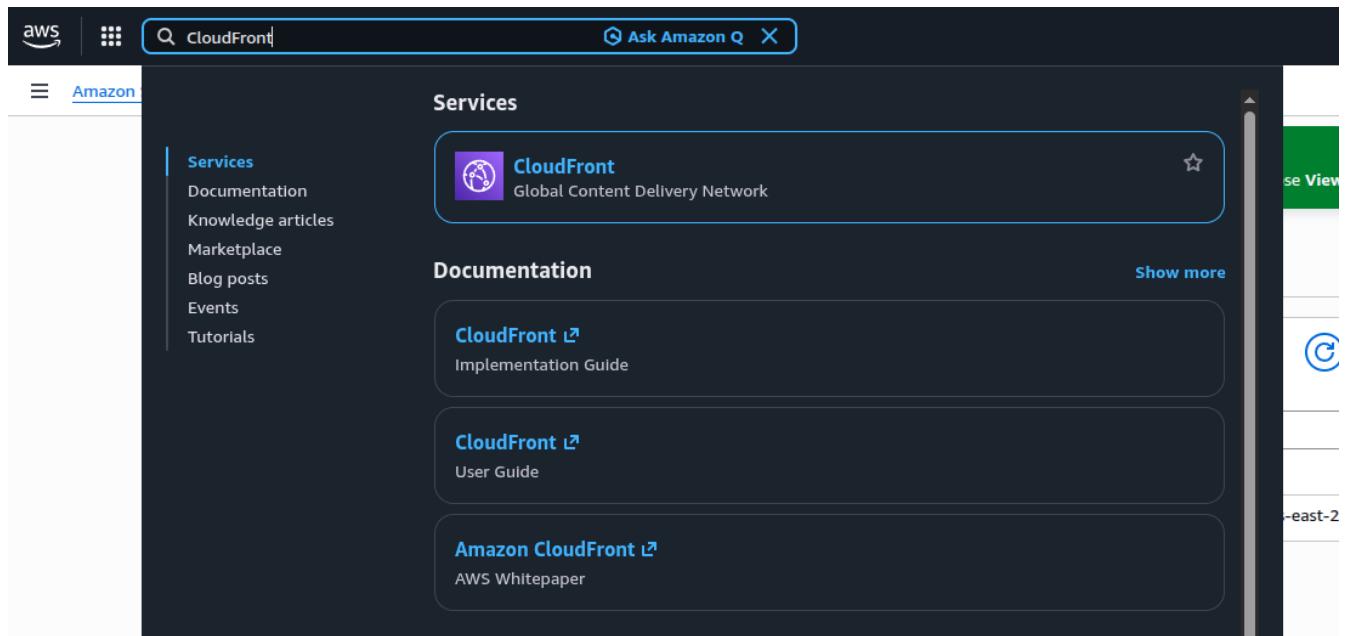
Click:

- **Create bucket**

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

The screenshot shows the AWS S3 console. At the top, a green banner displays the message: "Successfully created bucket 'my-sudoku-web'. To upload files and folders, or to configure additional bucket settings, choose View details." Below the banner, there are two tabs: "General purpose buckets" (which is selected) and "All AWS Regions". On the left, there's a sidebar with "Directory buckets". The main area shows a table for "General purpose buckets (1)". The table has columns for "Name", "AWS Region", and "Creation date". One row is listed: "my-sudoku-web" (US East (Ohio) us-east-2, February 8, 2026, 19:38:14 (UTC-05:00)). To the right of the table are three cards: "Account snapshot" (updated daily), "External access summary" (updated daily), and "Storage Lens" (provides visibility into storage usage and activity trends).

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:

A screenshot of the AWS CloudFront 'Distributions' list. The title 'Distributions (0)' is at the top, followed by a search bar and a filter dropdown set to 'All distributions'. There are columns for ID, Status, Description, Type, Domain name (standard), Alternate domain names, Origins, and Pricing plan. A large button labeled 'Create distribution' is prominently displayed. The message 'No distributions' and 'You don't have any distributions.' is shown below the table.

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

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

Step 1

- Get started
- Step 2
- Step 3
- Step 4
- Step 5
- Review and create

Get started

Connect your websites, apps, files, video streams, and other content to CloudFront. We optimize the performance, reliability, and security for your web traffic.

Distribution options

Distribution name

Name will be stored as a tag on the resource. You can change the name, or more tags, later.

Description - optional

Distribution type

Single website or app
Choose if each website or application will have a unique configuration.

Multi-tenant architecture - New
Choose when you have multiple domains that need to share configurations. This is a common architecture for SaaS providers.

Domain

Route 53 managed domain - optional

Enter a domain that's already registered with Route 53 in your AWS account. CloudFront will provision a TLS certificate for you. If you have a domain from a different DNS provider, skip this step and configure your domain later.

Check domain

Tags - optional

Cancel Next

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:

Specify origin

Origin type

Your origin is where your content (such as a website or app) lives. CloudFront works with AWS-based origins and origins hosted on other cloud providers.

Origin type

Amazon S3

Deliver static assets like files and images, statically generated websites or single page applications (SPA).

Elastic Load Balancer

Deliver applications hosted behind ELB such as dynamic websites, web services, and APIs.

API Gateway

Deliver API endpoints for REST APIs hosted on API Gateway.

Elemental MediaPackage

Deliver end-to-end live events or video on demand (VOD).

VPC origin

Deliver applications and content hosted within private VPCs, such as EC2 instances and Application Load Balancers.

Other

Refer to any AWS or non-AWS origin through its publicly resolvable URL.

Origin

S3 origin

Choose an AWS origin, or enter your origin's domain name. [Learn more ↗](#)

example-bucket.s3.us-east-1.amazonaws.com

[Browse S3](#)

Origin path - optional

The directory path within your origin where your content is stored. [Learn more ↗](#)

/path

Settings Info

CloudFront provides default origin and cache settings based on what origin you selected. [View default settings for S3 ↗](#)

Allow private S3 bucket access to CloudFront Info

CloudFront will update your S3 bucket policy to allow CloudFront to access your S3 bucket. The policy allows CloudFront to access the bucket only when the request is on behalf of the CloudFront distribution that contains the S3 origin.

Allow private S3 bucket access to CloudFront - Recommended

Origin settings

Origin settings control how CloudFront connects to the specified origin.

Use recommended origin settings

Customize origin settings

Cache settings

Cache settings determine when CloudFront serves cached content and when it fetches new content from the origin.

Use recommended cache settings tailored to serving S3 content

Customize cache settings

[Cancel](#)

[Previous](#)

[Next](#)

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

Select S3 location X

Buckets (1) Last updated
February 8, 2026, 08:52 PM GMT-5 

Bucket name	Creation date
my-sudoku-web	Mon, 09 Feb 2026 01:41:56 GMT

my-sudoku-web Cancel Choose

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:

Enable security

Web Application Firewall (WAF) Info

Enable security protections
Keep your application secure from the most common web threats and security vulnerabilities using AWS WAF. Blocked requests are stopped before they reach your web servers.

Do not enable security protections
Select this option if your application does not need security protections from AWS WAF.

Included security protections

- Protect against the most common vulnerabilities found in web applications.
- Protect against malicious actors discovering application vulnerabilities.
- Block IP addresses from potential threats based on Amazon Internal threat intelligence.

Use monitor mode
Count how many of your requests would be blocked by this WAF configuration. When ready, you can disable monitor mode to begin blocking requests.

Protection against Layer 7 DDoS attacks Recommended
Stops DDoS attacks within seconds. AWS learns your unique application patterns within minutes of activation, accurately distinguishing between attacks and natural traffic surges.

Price estimate

This AWS WAF configuration is estimated to cost \$14 for 10 million requests/month

Cancel Previous Next

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:

Review and create

General configuration

Distribution name sudoku-web	Description Sudoku Vite app (S3 + CloudFront)	Billing Pay-as-you-go (\$0/month)
---------------------------------	--	--------------------------------------

Origin

Because you granted CloudFront access to your origin, CloudFront can write and update S3 bucket policies that restrict access to your S3 origin to CloudFront.			
S3 origin my-sudoku-web.s3.us-east-2.amazonaws.com	Origin path -	Grant CloudFront access to origin Yes	Enable Origin Shield No
Connection attempts 3	Connection timeout 10		

Cache settings

CloudFront will apply default cache settings tailored to serving content from a S3 origin. You can customize settings after you create your distribution.		
---	--	--

Security

Security protections None	Use monitor mode No	Use existing WAF configuration No
------------------------------	------------------------	--------------------------------------

[Cancel](#) [Previous](#) [Create distribution](#)

Next, you will see a screen like this one:

sudoku-web Standard [View metrics](#)

Details

Distribution domain name d1td03ke8lbqp6.cloudfront.net	Billing C	ARN arn:aws:cloudfront::286119371384:distribution/E SHIPEWHEGNGU	Last modified February 9, 2026 at 1:56:06 AM UTC
---	--------------	---	---

[General](#) [Security](#) [Origins](#) [Behaviors](#) [Error pages](#) [Invalidations](#) [Logging](#) [Tags](#)

Error pages (0)

HTTP error code	▲ Minimum TTL (seconds)	▼ Response page path	▼ HTTP response code
No error pages You don't have any error pages.			
Create custom error response			

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.

The screenshot shows the AWS S3 console interface. At the top, there are two tabs: "General purpose buckets" (which is selected) and "Directory buckets". Below the tabs, a header bar includes "All AWS Regions", a "Copy ARN" button, an "Empty" button, a "Delete" button, and a prominent orange "Create bucket" button. The main area displays a table titled "General purpose buckets (1)". The table has a single row with the following data:

- Name: my-sudoku-web
- AWS Region: US East (Ohio) us-east-2
- Creation date: February 8, 2026, 20:41:56 (UTC-05:00)

Below the table, there is a search bar with the placeholder "Find buckets by name" and navigation controls for pages 1 and 2.

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:

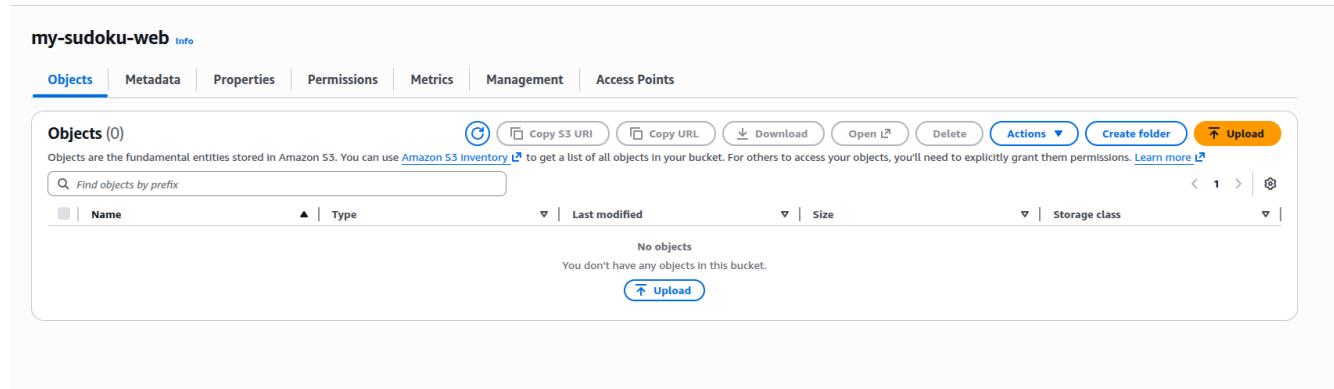
```
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react";

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:

Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDKs or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose Add files or Add folder.

Files and folders (18 total, 30.0 MB)

All files and folders in this table will be uploaded.

Find by name

[Remove](#)

[Add files](#)

[Add folder](#)

<input type="checkbox"/>	Name	Folder	Type	Size	<input type="checkbox"/>
<input type="checkbox"/>	EASY.mp3	music/	audio/mpeg	1.9 MB	
<input type="checkbox"/>	START.mp3	music/	audio/mpeg	3.0 MB	
<input type="checkbox"/>	MEDIUM.mp3	music/	audio/mpeg	2.3 MB	
<input type="checkbox"/>	SAMURAI.mp3	music/	audio/mpeg	2.0 MB	
<input type="checkbox"/>	HARD.mp3	music/	audio/mpeg	2.4 MB	
<input type="checkbox"/>	VICTORY.mp3	music/	audio/mpeg	2.5 MB	
<input type="checkbox"/>	index-CxxniBdH.css	assets/	text/css	21.9 KB	
<input type="checkbox"/>	index-bWu2qj_W.js	assets/	text/javascript	225.9 KB	
<input type="checkbox"/>	index.html	-	text/html	462.0 B	
<input type="checkbox"/>	vite.svg	-	image/svg+xml	1.5 KB	

Destination Info

Destination

[s3://my-sudoku-web](#)

Destination details

Bucket settings that impact new objects stored in the specified destination.

Permissions

Grant public access and access to other AWS accounts.

Properties

Specify storage class, encryption settings, tags, and more.

[Cancel](#)

[Upload](#)

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:

Distributions (1) Info



[Enable](#)

[Disable](#)

[Delete](#)

[Create distribution](#)

< 1 > |

<input type="checkbox"/>	ID	Status	Description	Type	Domain name (standard)	Alternate domain names	Origins	Pricing plan
<input type="checkbox"/>	ESHIPWHEGNGU	Enabled	Sudoku Vite app (S3 + Clou...	Standard	d1tod5ke8lbqp6.cloud...	-	my-sudoku-web.s3.us-east-2.amazonaws.com	Pay-as-you-go

Click on the one which was meant for the game.

Finally, you will see this:

sudoku-web Standard

[View metrics](#)

Details

Distribution domain name d1tod3ke8lbqp6.cloudfront.net	Billing €	ARN arn:aws:cloudfront::286119371384:distribution/E SHIPEWHEGNGU	Last modified February 9, 2026 at 2:01:54 AM UTC
---	--------------	---	---

General Security Origins Behaviors Error pages Invalidations Logging Tags

Settings

Name: sudoku-web [Edit](#)

Description: Sudoku Vite app (S3 + CloudFront)

Price class: Use all edge locations (best performance)

Supported HTTP versions: HTTP/2, HTTP/1.1, HTTP/1.0

Alternate domain names: [Add domain](#)

Standard logging: Off

Cookie logging: Off

Default root object: -

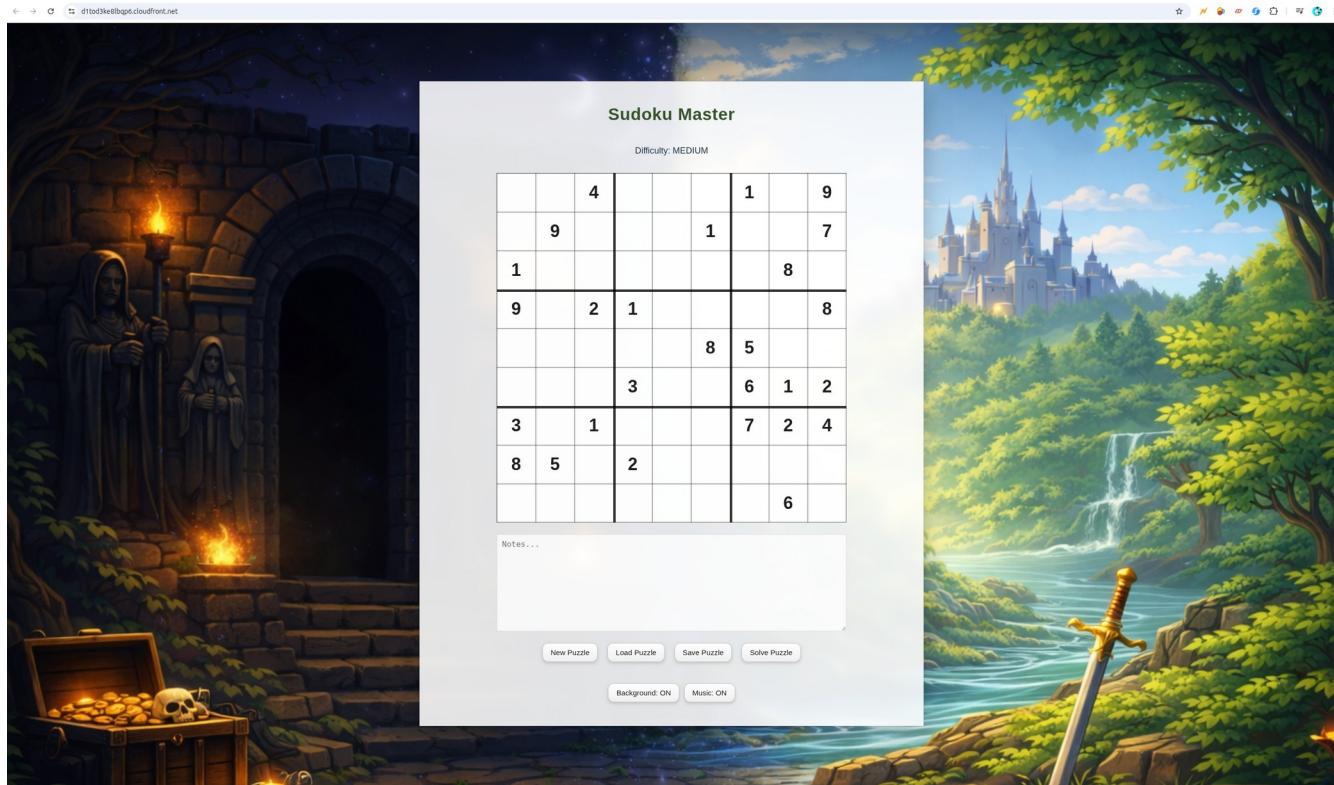
Continuous deployment [Info](#)

[Create staging distribution](#)

Copy the distribution name, in my case “[d1tod3ke8lbqp6.cloudfront.net](#)” and go to

<https://d1tod3ke8lbqp6.cloudfront.net>

The game is ON, HAPPY GAMING !

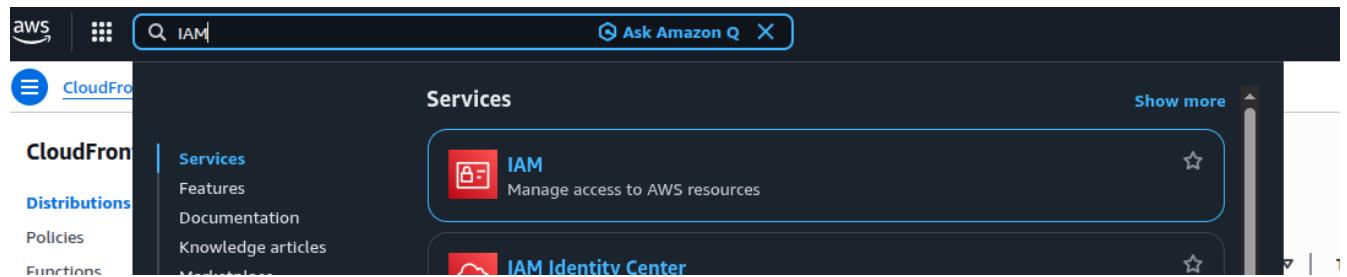


Please note, I have removed this so I do not get charged. I have done another pipeline deployment on [github.io](#), which was free, see: <https://olejardamir.github.io/sudoku/>

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**.

Add Identity provider Info

Provider details

Provider type Info

SAML
Establish trust between your AWS account and a SAML 2.0 compatible Identity Provider such as Shibboleth or Active Directory Federation Services.

OpenID Connect
Establish trust between your AWS account and Identity Provider services, such as Google or Salesforce.

Provider URL
Specify the secure OpenID Connect URL for authentication requests.

Maximum 255 characters. URL must begin with "https".

Audience Info
Specify the client ID issued by the Identity provider for your app.

Maximum 255 characters. Use alphanumeric or "-_./" characters.

Add tags - optional Info
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.
No tags associated with the resource.

Add new tag
You can add up to 50 more tags.

Cancel **Add provider**

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

token.actions.githubusercontent.com added. You must assign an IAM role to start using this provider.

Have you considered using AWS IAM Identity Center?
AWS IAM Identity Center makes it easy to centrally manage access to multiple AWS accounts and provide users with single sign-on access to all their assigned accounts from one place. With IAM Identity Center, you can create and manage user identities in IAM Identity Center or easily connect to your existing SAML 2.0 compatible identity provider. [Learn more](#)

Identity providers (1) Info

Use an identity provider (idp) to manage your user identities outside of AWS, but grant the user identities permissions to use AWS resources in your account.

Group name	Type	Creation time
token.actions.githubusercontent.com	OpenID Connect	Now

Delete **Add provider**

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.githubusercontentcontent.com
- Audience:
sts.amazonaws.com
- Organization:
YOUR GITHUB USERNAME

Select trusted entity [Info](#)

Trusted entity type

AWS service
Allows AWS services like EC2, Lambda, or others to perform actions in this account.

AWS account
Allows users in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

SAML 2.0 federation
Allows users federated with SAML 2.0 from a corporate directory to perform actions in this account.

Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

Identity provider
 [Create new](#) [?](#)

Audience

GitHub organization

Must be maximum 39 characters and can only contain letters, numbers, -, _.

GitHub repository - optional

Must be maximum 100 characters and can only contain printable characters.

GitHub branch - optional

Must be maximum 255 characters and can only contain printable characters.

[Cancel](#) [Next](#)

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

The screenshot shows the AWS IAM Roles page. At the top, a green banner indicates "Role github-actions-sudoku-deploy created." Below this, a table lists five roles:

Role name	Trusted entities	Last activity
AWSServiceRoleForResourceExplorer	AWS Service: resource-explorer-2 (Service)	35 minutes ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linker)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service)	-
github-actions-sudoku-deploy	Identity Provider: arn:aws:sam::2861	-

Go to Trust relationships → Edit trust policy

The screenshot shows the "Edit trust policy" page for the "github-actions-sudoku-deploy" role. The tab "Trust relationships" is selected. The page displays the current trust policy JSON:

```
1< {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Principal": {  
7         "Federated": "arn:aws:iam::286119371384:oidc-provider/token.actions.githubusercontent.com"  
8       },  
9       "Action": "sts:AssumeRoleWithWebIdentity",  
10      "Condition": {  
11        "StringEquals": {  
12          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"  
13        },  
14        "StringLike": {  
15          "token.actions.githubusercontent.com:sub": [  
16            "repo:olejardamir/*",  
17            "repo:olejardamir/*"  
18          ]  
19        }  
20      }  
21    }  
22  ]  
23 }
```

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

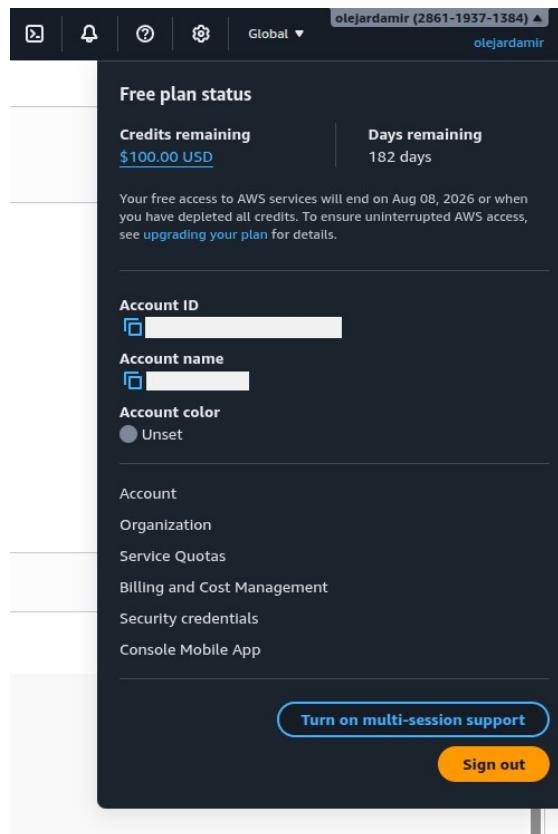
```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "Federated":  
          "arn:aws:iam::<ACCOUNT_ID>:oidc-provider/token.actions.githubusercontent.com"  
      },  
      "Action": "sts:AssumeRoleWithWebIdentity",  
      "Condition": {  
        "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

The screenshot shows the GitHub repository settings page for 'Actions'. The left sidebar has a 'General' tab selected under 'Actions and automation'. Other tabs include 'Access', 'Collaborators', 'Moderation options', 'Code and automation' (with 'Branches', 'Tags', 'Rules', 'Actions', 'Models', 'Webhooks', 'Copilot', 'Environments', 'Codespaces', 'Pages'), 'Security' (with 'Advanced Security', 'Deploy keys', 'Secrets and variables' which is currently selected), 'Actions', 'Codespaces', and 'Dependabot'. Under 'Integrations', there are 'GitHub Apps' and 'Email notifications'. The main content area is titled 'Actions secrets and variables'. It contains two sections: 'Environment secrets' (which says 'This environment has no secrets.' and has a 'Manage environment secrets' button) and 'Repository secrets' (which says 'This repository has no secrets.' and has a 'New repository secret' button). A 'Preview' button is visible near the 'Actions' tab in the sidebar.

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

Name *

AWS_ROLE_ARN

Secret *

arn:aws:iam::286119371384:policy/github-actions-sudoku-deploy-policy

Add secret

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.

Repository secrets

New repository secret

Name	Last updated		
AWS_REGION	now		
AWS_ROLE_ARN	1 minute ago		
CLOUDFRONT_DISTRIBUTION_ID	now		
S3_BUCKET	now		

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 now fully automated with AWS and Github CI/CD pipeline.