

<input type="checkbox"/>	Name	Type	Last modified	<input type="checkbox"/>	Size
<input type="checkbox"/>	CustomErrors /	Folder	-		

## Project 1: Working with Amazon CloudFront

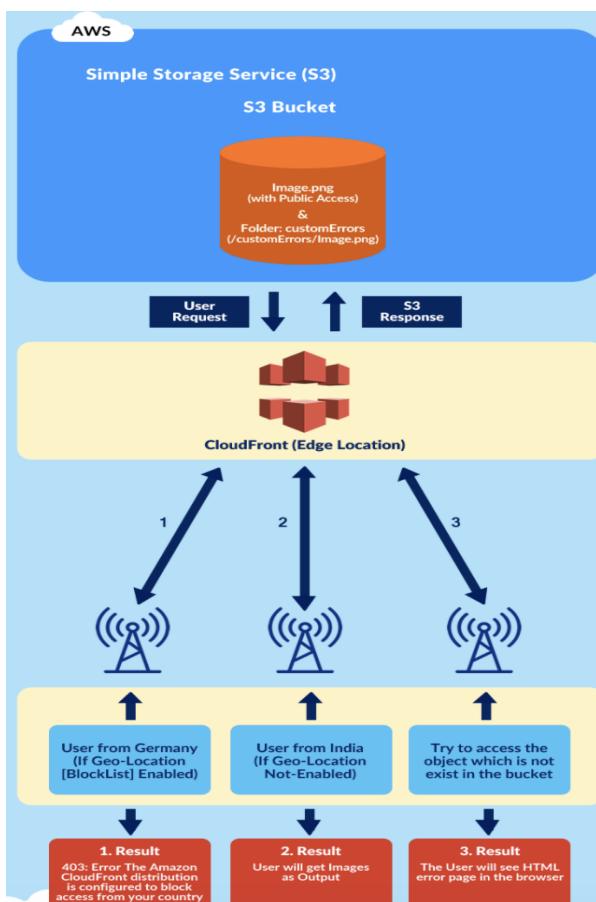
### What is Amazon CloudFront?

- Amazon CloudFront is a content delivery network offered by AWS. Content Delivery networks provide globally-distributed network of proxy servers which cache content i.e. videos or bulky media, more locally to consumers, thus improving access speed for downloading the content.

### Objective of this Project

- We will be demonstrating how a CloudFront Distribution is created. In order for it to work, we will be distributing a publicly accessible image stored in the S3 bucket.

### Architecture



	Name	Type	Last modified	Size
<input type="checkbox"/>	CustomErrors /	Folder	-	

## Step 1: Creating an S3 Bucket

We just created an S3 bucket with the general configuration settings and made sure it was publicly accessible.

### General configuration

Bucket name

mys3bucket543

Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#) 

AWS Region

US East (N. Virginia) us-east-1

*Copy settings from existing bucket - optional*

Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

	Name	Type	Last modified	Size
<input type="checkbox"/>	CustomErrors /	Folder	-	

### 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.

## Step 2: Uploading a file to an S3 bucket

We then upload a file into the S3 bucket.

The screenshot shows the AWS S3 console interface for uploading files. At the top, there is a table header with columns: Name, Type, Last modified, and Size. Below the header, a folder named "CustomErrors/" is listed as a Folder. The main area is titled "Upload" and contains instructions: "Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more" with a link icon. A dashed blue border indicates where files can be dragged and dropped. Below this, a section titled "Files and folders (1 Total, 60.3 KB)" shows one item: "Harry Potter.jpeg" (image/jpeg, 60.3 KB). There are buttons for "Remove", "Add files", and "Add folder". A search bar with placeholder text "Find by name" and navigation arrows (< 1 >) are also present. The bottom section is titled "Destination" and shows the destination as "s3://mys3bucket543".

Upload of the file being successful

The screenshot shows the AWS S3 console displaying the upload summary. It includes a "Summary" section with three status boxes: "Destination s3://mys3bucket543" (Succeeded, 1 file, 60.3 KB (100.00%)), "Succeeded" (1 file, 60.3 KB (100.00%)), and "Failed" (0 files, 0 B (0%)).

## Step 3: Creating Custom Error Pages

The objective of creating custom error pages for CloudFront to return when origin returns HTTP 400 or 500 errors.

We start with creating a folder, naming it Custom Errors

We then create an error.html file:

	Name	Type	Last modified	Size
□	CustomErrors /	Folder	-	

<html>

<h1>

**This is an Error Page**

</h1>

</html>

And also a block.html file

```
<html>
  <h1>
    This content is blocked in your location!!!!
  </h1>
</html>
```

	Name	Type	Last modified	Size		
□	CustomErrors /	Folder	-			
After creating the two files, we upload error.html on the S3 bucket and upload block.html on the Custom Errors Folder.						
<b>Summary</b>						
Destination	Succeeded		Failed			
s3://mys3bucket543/CustomErrors /	✓ 1 file, 447.0 B (100.00%)		✗ 0 files, 0 B (0%)			
<b>Files and folders</b>		Configuration				
<b>Files and folders (1 Total, 447.0 B)</b>						
<input type="text"/> Find by name						
Name	Folder	Type	Size	Status		
block.html.rtf	-	text/rtf	447.0 B	✓ Succeeded		
□	CustomErrors /	Folder	-	-		
□	error.html.rtf	rtf	June 22, 2021, 08:19:45 (UTC-07:00)	994.0 B Standard		
□	Harry Potter.jpeg	jpeg	June 22, 2021, 08:07:42 (UTC-07:00)	60.3 KB Standard		

## Step 4: Making the objects public

When we first click on the image URL, you will see an Access Denied message.

```
<Error>
<Code>AccessDenied</Code>
<Message>Access Denied</Message>
<RequestId>8VJHE1JP6QZYB4P2</RequestId>
<HostId>9+Ohg/BhfO8CV51RytuLgXScJfwLoqUw0e09tRgp7243HdGRWquoAFaX4u3zEyL2pIx6eT/tYx4=</HostId>
</Error>
```

We immediately update the bucket policy with this, written in JSON:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
    }
  ]
}
```

	Name	Type	Last modified	Size
CustomErrors /	Folder	-		

```

    "Action": "s3>ListBucket",
    "Resource": "arn:aws:s3:::mys3bucket543"
},
{
    "Effect": "Allow",
    "Principal": {
        "AWS": "*"
    },
    "Action": [
        "s3:GetObject",
        "s3:PutObject"
    ],
    "Resource": "arn:aws:s3:::mys3bucket543/*"
}
]
}

```

⌚ Successfully edited bucket policy.

Now that policy has been editing, we click on the URL of the image and the result is this: IT'S WORKING!!!!



## Step 5: Creating A CloudFront Distribution

We go on CloudFront and Create A Distribution First where the origin is from the S3 Bucket.

<input type="checkbox"/>	Name	Type	Last modified	<input type="checkbox"/>	Size
<input type="checkbox"/>	CustomErrors /	Folder	-		

CloudFront Distributions									
<a href="#">Create Distribution</a>		<a href="#">Distribution Settings</a>		<a href="#">Delete</a>	<a href="#">Enable</a>	<a href="#">Disable</a>			
Viewing :		Any Delivery Method	Any State	Viewing 1 to 1 of 1 Items					
Delivery Method	ID	Domain Name	Comment	Origin	CNAMEs	Status	State	Last Modified	
<input type="checkbox"/> Web	ESKEHEIGRAVXK	d1p0kc9ww07idq.	-	mys3bucket	-	Deployed	Enabled	2021-06-22 08:39	

## Step 6: Accessing Image Through CloudFront

We then type in the URL found under domain name and attach the file name on the URL.

**Domain Name** d1p0kc9ww07idq.cloudfront.net

The end result is this. The image loaded faster than what it was supposed to be.



## Step 7: Customizing an Error Page

We then create a customized error page using the error.html file that can be displayed if an error on the CloudFront URL were to occur.

We test it on the browser and see if it works or not by utilizing a random URL page. The end result is this.

<input type="checkbox"/>	Name	Type	Last modified	<input type="checkbox"/>	Size
<input type="checkbox"/>	CustomErrors /	Folder	-		

# This is an Error Page

## Step 8: Restricting the Geographic Distribution of Your Content

If we need to prevent users in selected countries from accessing content, we can either whitelist or blacklist content by using restrictions.

Here's how this process is done.

### Geo-Restriction Settings

Enable Geo-Restriction	<input checked="" type="radio"/> Yes <input type="radio"/> No	
Restriction Type	<input type="radio"/> Whitelist <input checked="" type="radio"/> Blacklist	
Countries		
	AF -- AFGHANISTAN AX -- ALAND ISLANDS AL -- ALBANIA DZ -- ALGERIA AS -- AMERICAN SAMOA AD -- ANDORRA	<input type="button" value="Add &gt;&gt;"/> <input type="button" value="&lt;&lt; Remove"/>
	US -- UNITED STATES	

We will then test the following to see how it works. The end result is a 403 Error.

## 403 ERROR

**The request could not be satisfied.**

---

The Amazon CloudFront distribution is configured to block access from your country. We can't connect to the server for this app or website at this time.'