Table of contents

- Table of contents
- What is CORS and why it is needed :
    - Pre-requisites
    - Step 1
    - Step 2
    - Setp 3
    - Points to consider

What is CORS and why it is needed :

- S3 CORS stands for cross origin resource sharing .
- By default browsers block accessing resources from another domain
- S3 CORS allows us to share the resources from one bucket to another bucket hosting a static website .

Pre-requisites

- Since we need to test CORS , we need 2 buckets(A and B) having static website enabled and public access enabled .
- We will need 3 html files
    - index.html :- This will act as a home page where we will be testing if we can fetch object from another bucket
        - Content for the same can be as below to begin with

```
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js">
</script>
</head>
<h1>This is the index.html being called</h1>
<div id="loadDiv"></div>
<html>
<script type="text/javascript">
    $("#loadDiv").load("external.html");
</script>
```

```
- As seen above , the index.html is calling another file in its division called
external.html. Let us create that as well
```

- external.html :- This will act as a external source . it can have any simplified content that we should be able to see on index.html

```
<html>
<h5>this is from external.html</h5>
<html>
```

- error.html :- this is our usual file which we use to test if the home page is reachable

```
<html>
<h1>ERRORRRRRRRRRR!!!!!!!!!!!!</h1>
<html>
```

Step 1

- To begin with , we will keep all 3 files inside one bucket i.e. bucket A . We will test if the content of external.html is being fetched in index.html
- Upload all 3 files in one bucket(A) . Make sure bucket and these 3 objects are made public.
- Make index.html as the home page and error.html as error page while enabling static website hosting .
- Check the endpoint in the browser to make sure the home page is visible and we are able to see the content from external.html as well
- Ideally the content should be visile as all files are in the same bucket i.e. domain

Step 2

- Now that we have established that the if the files are in same bucket or domain , they are accessible. We need to test if we can access it from other bucket or domain
- In order to do it , upload external.html file another bucket i.e. bucket B . make sure it is public
- Now we need to make changes in the index.html file in our original bucket i.e. bucket A .

```
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js">
</script>
</head>
<h1>This is the index.html being called</h1>
<div id="loadDiv"></div>
<html>
<script type="text/javascript">
    $("#loadDiv").load("http://bucketB.s3-website.ap-south-
1.amazonaws.com/external.html");
</script>
```

- Observe that the filename of external.html has been replaced with the entire url of the object from another bucket
- Once the changes are made, reupload the file in bucket A to make sure the changes are reflected.

- Try accessing the website using the endpoint from Bucket A
- Observe that the content from external.html is not visible anymore
- It establishes the point made earlier that browser blocks cross domain resource sharing
- This can also be estalished by right clickin on the page and clicking on "inspect" under console tab
- We should ideally see " blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource."

Setp 3

- Since we have established now that the resource sharing is blocked by default , let us check how do we enable it
- Navigate to bucket B
- Under the permissions , we have CORS tab .
- Similar to bucket policy , we can paste a policy here which dica

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://bucketA.s3-website-us-east1.amazonaws.com"
        ],
        "ExposeHeaders": []
    },
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://bucketA.s3-website-us-east1.amazonaws.com"
        ],
        "ExposeHeaders": []
    },
    {
        "AllowedHeaders": [],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": []
    }
]
```

- Observe that in allowed orign we have put the endpoint of bucket A
- Once done check the the endpoint of bucket A in browser , the external.html file should be accessible

Points to consider

- It is easy to get confused with bucket A and bucket B , remember we will be always testing the access using endpoint of bucket A . Bucket B is there just to host external.html file and CORS file
- All 3 files should always be public in order to test the above scenario
- We are making file level changes in only one file i.e. index.html. Once the changes are done, we have to reupload the file in bukcet A to make the changes reflect
- While using the content above , make sure you are replacing bucket names in the policy wherever needed