CLOUDFRONT SIGNED COOKIES

Step 1. S3 Bucket Creation

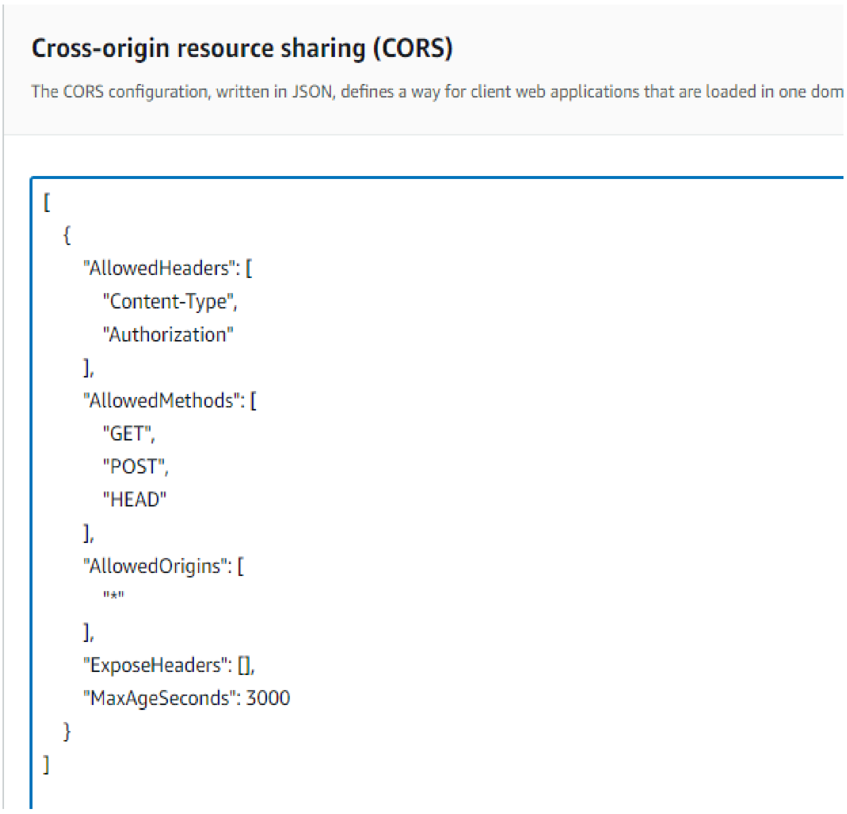
1. Sign in to the AWS Management Console and open the S3 bucket
2. In the navigation menu, choose **Buckets**
3. Choose **Create bucket**

   a. For bucket type, select default type **General Purpose**
   b. For Bucket name, give the name to identify the bucket
   c. For Object Ownership, select ACLs enabled
   d. For Block Public Access settings for this bucket , turn on **Block all public access**
   e. Remaining option keep the same

   When finished , choose **Create bucket**

4. S3 **Cross-origin resource sharing (CORS)** configuration

   Add the following CORS configuration for the s3 bucket you created



Cross-origin resource sharing (CORS)

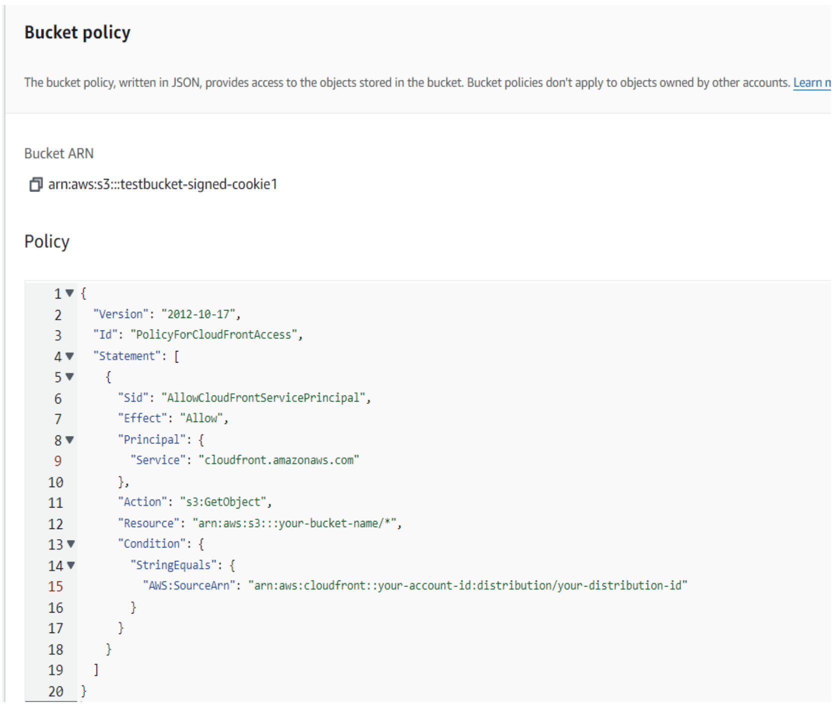
The CORS configuration, written in JSON, defines a way for client web applications that are loaded in one dom

```
[
  {
    "AllowedHeaders": [
      "Content-Type",
      "Authorization"
    ],
    "AllowedMethods": [
      "GET",
      "POST",
      "HEAD"
    ],
    "AllowedOrigins": [
      "*"
    ],
    "ExposeHeaders": [],
    "MaxAgeSeconds": 3000
  }
]
```

5. Create a Folder in S3 bucket and upload videos (Including m3u8 file and different resolution videos)

6. **Bucket Policy**

Add the bucket name in the place your-bucket-name , account-id and distribution id after creation of cloudfront



Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn n

Bucket ARN

arn:aws:s3:::testbucket-signed-cookie1

Policy

```
{
  "Version": "2012-10-17",
  "Id": "PolicyForCloudFrontAccess",
  "Statement": [
    {
      "Sid": "AllowCloudFrontServicePrincipal",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudfront.amazonaws.com"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::your-bucket-name/*",
      "Condition": {
        "StringEquals": {
          "AWS:SourceArn": "arn:aws:cloudfront::your-account-id:distribution/your-distribution-id"
        }
      }
    }
  ]
}
```

Step 2 . CloudFront Configuration and Set up

a) Create a key pair for a trusted key group

To create a key pair for a trusted key group, perform the following steps:

- Create the public–private key pair.

The following command uses OpenSSL to generate an RSA key pair with a length of 2048 bits and save to the file named private_key.pem

```
openssl genrsa -out private_key.pem 2048
```

The resulting file contains both the public and the private key. The following command extracts the public key from the file named private_key.pem

```
openssl rsa -pubout -in private_key.pem -out public_key.pem
```

Upload the public key (in the public_key.pem file) later, in the following procedure.

b) Upload the public key to CloudFront

1. Sign in to the AWS Management Console and open the CloudFront console.

2. In the navigation menu, choose **Public keys**.

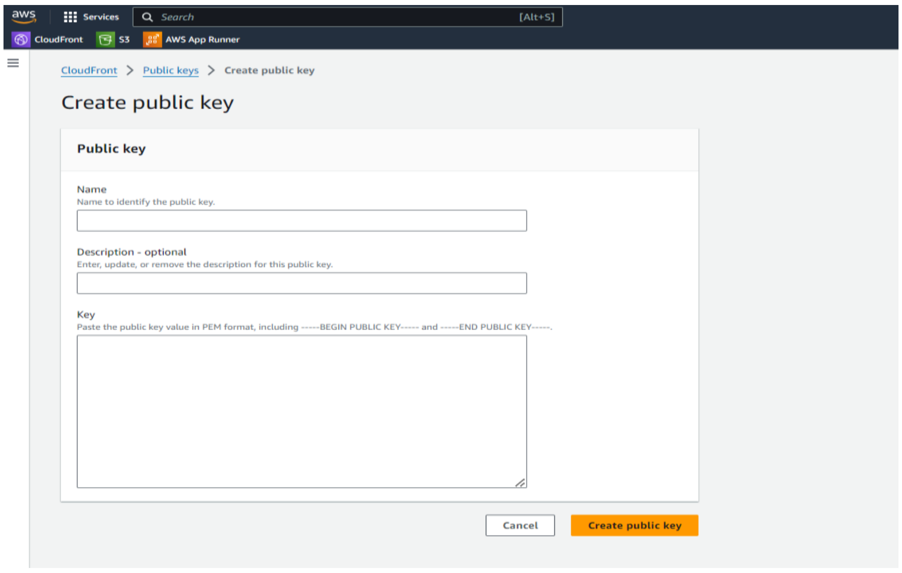
3. Choose **Create public key**.

4. In the **Create public key** window, do the following:

    a. For **Key name**, type a name to identify the public key.

    b. For **Key value**, paste the public key. If you followed the steps in the preceding procedure, the public key is in the file named public_key.pem. To copy and paste the contents of the public key

    c. (Optional) For **Comment**, add a comment to describe the public key.



Create public key

Public key

Name
Name to identify the public key.

Description - optional
Enter, update, or remove the description for this public key.

Key
Paste the public key value in PEM format, including -----BEGIN PUBLIC KEY----- and -----END PUBLIC KEY-----.

When finished, choose **Add**.

5. Record the public key ID. You use it later when you create signed cookies, as the value of the Key-Pair-Id field.

c) Add the public key to a key group

1. Open the CloudFront console.

2. In the navigation menu, choose **Key groups**.

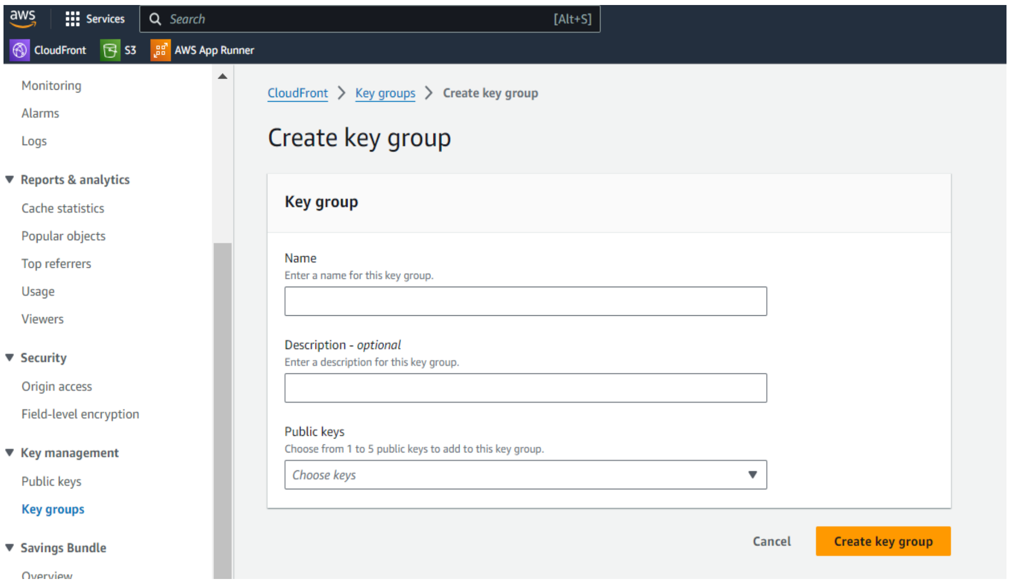
3. Choose **Add key group**.

4. On the **Create key group** page, do the following:

    a. For **Key group name**, type a name to identify the key group.

    b. (Optional) For **Comment**, type a comment to describe the key group.

    c. For **Public keys**, select the public key to add to the key group, then choose **Add**. Repeat this step for each public key that you want to add to the key group.



Create key group

Key group

Name
Enter a name for this key group.

Description - *optional*
Enter a description for this key group.

Public keys
Choose from 1 to 5 public keys to add to this key group.
Choose keys

Cancel | Create key group

5. Choose **Create key group**.

6. Record the key group name. You use it later to associate the key group with a cache behavior in a CloudFront distribution. (In the CloudFront API, you use the key group ID to associate the key group with a cache behavior.)

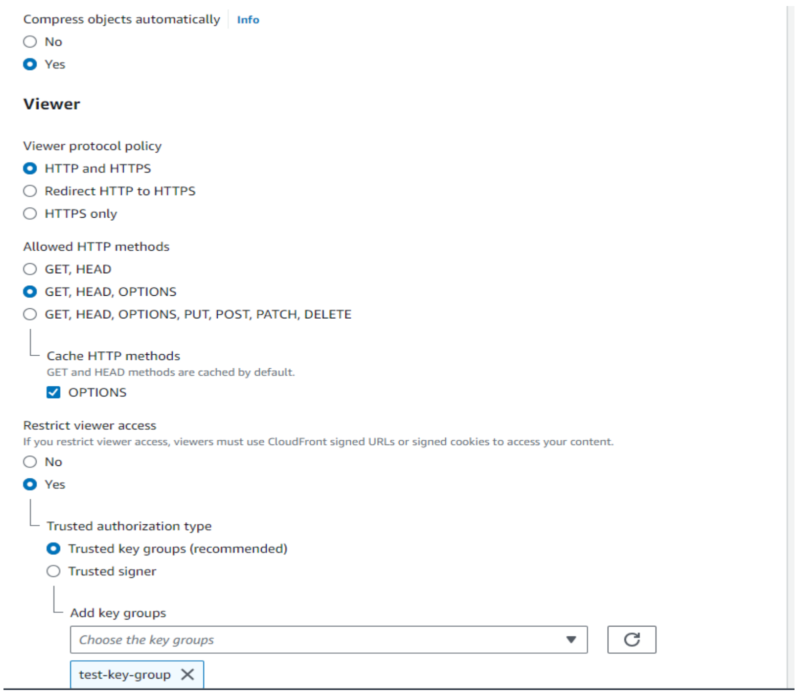
d) CloudFront Distribution Creation

1. Sign in to the AWS Management Console and open the cloudfront console

2. In the navigation menu, choose **Distributions**

3. Choose **Create distribution**

    a. For Origin domain, select the s3 bucket you are created
    b. (Optional) For Original Path, Enter a URL path to append to the origin domain name for origin requests.



Compress objects automatically Info
No [✓ Yes]

Viewer

Viewer protocol policy
[✓ HTTP and HTTPS] Redirect HTTP to HTTPS HTTPS only

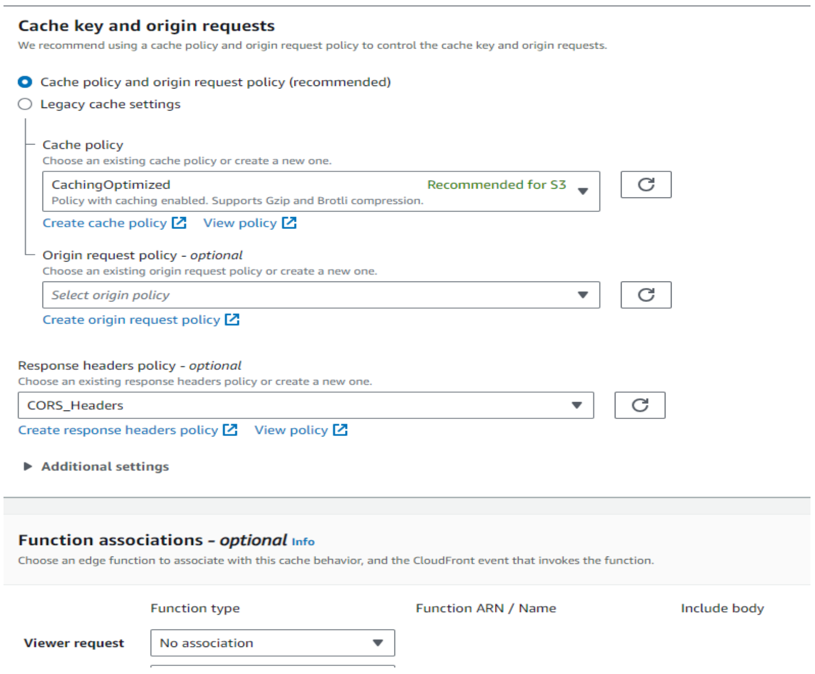
Allowed HTTP methods
GET, HEAD [✓ GET, HEAD, OPTIONS] GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

Cache HTTP methods
GET and HEAD methods are cached by default.
[✓ OPTIONS]

Restrict viewer access
If you restrict viewer access, viewers must use CloudFront signed URLs or signed cookies to access your content.
No [✓ Yes]

Trusted authorization type
[✓ Trusted key groups (recommended)] Trusted signer

Add key groups
Choose the key groups
test-key-group



Cache key and origin requests
We recommend using a cache policy and origin request policy to control the cache key and origin requests.

[✓ Cache policy and origin request policy (recommended)] Legacy cache settings

Cache policy
Choose an existing cache policy or create a new one.
CachingOptimized — Recommended for S3
Policy with caching enabled. Supports Gzip and Brotli compression.
Create cache policy | View policy

Origin request policy - *optional*
Choose an existing origin request policy or create a new one.
Select origin policy
Create origin request policy

Response headers policy - *optional*
Choose an existing response headers policy or create a new one.
CORS_Headers
Create response headers policy | View policy

Additional settings

Function associations - *optional* Info
Choose an edge function to associate with this cache behavior, and the CloudFront event that invokes the function.

	Function type	Function ARN / Name	Include body
Viewer request	No association		

c. For name, enter the name for this origin , default it will take the origin domain name
d. For Origin access, select Origin Access control Settings.

Click on the drop down and select the s3 bucket origin
e. (Optional) For Custom headers
- Add **Access-Control-Allow-Origin** for Header Name and
- Add * for value

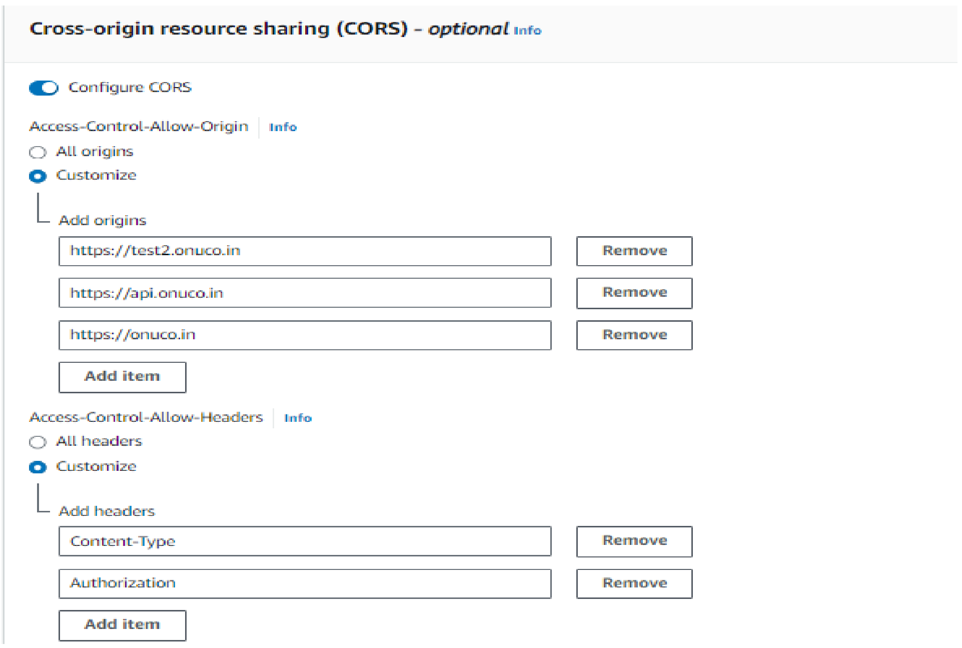
f. Keep Enable Origin field to **No**

Default cache behaviour

g. Follow the reference image for Viewer protocol policy, Allowed HTTP methods and Restrict viewer access.
For key group, select the key group previously created

h. For Response headers policy, Create response headers policy

* For Name, enter a name for the response headers policy
* (Optional) Description , enter a description for the response headers policy.



Cross-origin resource sharing (CORS) - *optional* Info

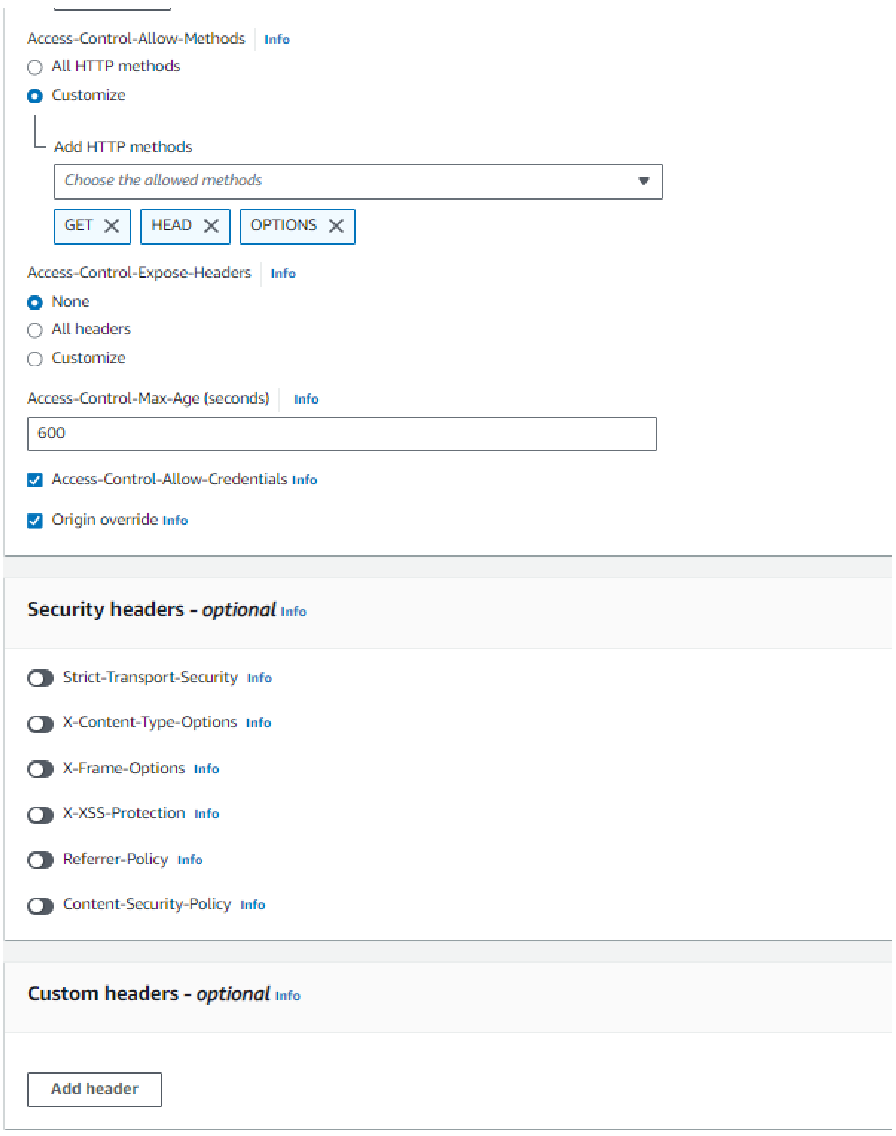
Configure CORS

Access-Control-Allow-Origin Info
All origins [✓ Customize]

Add origins
https://test2.onuco.in
https://api.onuco.in
https://onuco.in

Access-Control-Allow-Headers Info
All headers [✓ Customize]

Add headers
Content-Type
Authorization


Access-Control-Allow-Methods Info

All HTTP methods [✓ Customize]

Add HTTP methods: Choose the allowed methods — GET, HEAD, OPTIONS

Access-Control-Expose-Headers Info

[✓ None] All headers Customize

Access-Control-Max-Age (seconds) Info: 600

✓ Access-Control-Allow-Credentials Info

✓ Origin override Info

Security headers - *optional* Info

- Strict-Transport-Security Info
- X-Content-Type-Options Info
- X-Frame-Options Info
- X-XSS-Protection Info
- Referrer-Policy Info
- Content-Security-Policy Info

Custom headers - *optional* Info

Keep the other options to default, when finished , Click on Create.

* Then select the created policy for response header policy.

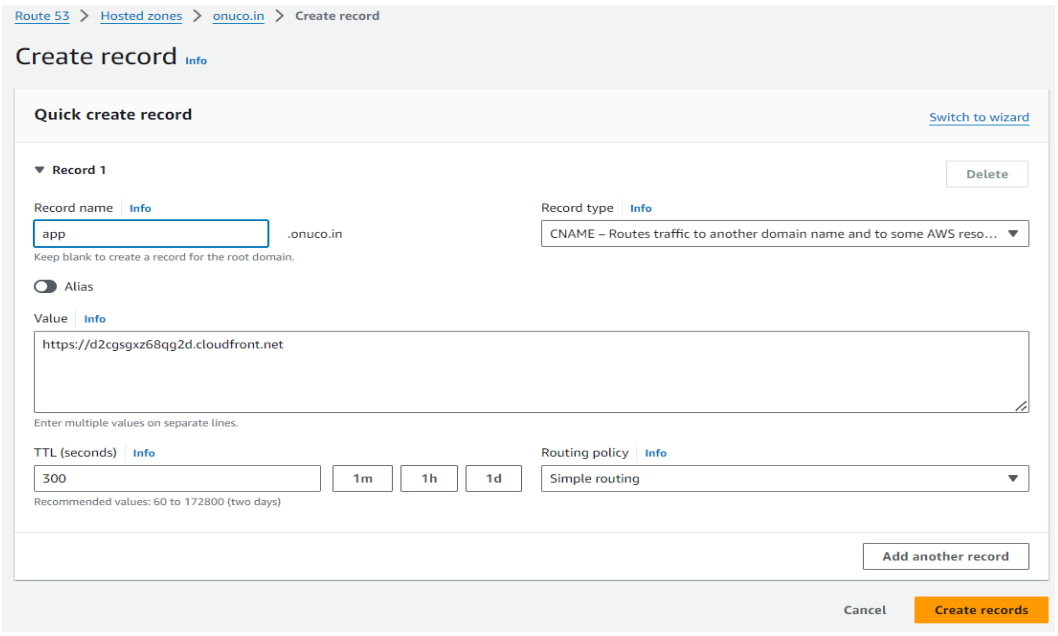
* Keep the Other settings to default

i. When finished, Click on Create Distribution.

Note the distribution ID and add the ID in s3 bucket policy.

e) Create alternate Domains for CloudFront

1. Sign in to the AWS Management Console and open the **Route 53**
2. In the navigation menu, choose **Hosted Zones**
3. Choose the zone you hosted your application (e.g., ‘Onuco.in’)
4. Choose **Create Record**



Record name: app .onuco.in

Keep blank to create a record for the root domain.

Record type: CNAME – Routes traffic to another domain name and to some AWS reso...

Alias: off

Value: https://d2cgsgxz68qg2d.cloudfront.net

Enter multiple values on separate lines.

TTL (seconds): 300 (1m, 1h, 1d)

Recommended values: 60 to 172800 (two days)

Routing policy: Simple routing

a. For Record name, enter the sub domain name (e.g., ‘app’ for ‘app.onuco.in’)

b. For Record Type, select the type of record i.e., CNAME

c. Value, enter CloudFront distribution domain (e.g., https://dg0abcdefg2e.cloudfront.net)

d. Keep the other settings to default.

When finished, Click on **Create record**

f) Add Alternate Domain for CloudFront Distribution

1. Open CloudFront
2. In the navigation menu, choose **Distributions**
3. Choose the s3 cloudfront distribution
4. In General tab, under settings click on **Edit**

   a. Price Class, select Use all edge locations (best performance)

   b. Alternate Domain Name (CNAME) , Click on add item

   c. Enter the Domain you previously created for cloudfront

   d. Custom SSL certificate, select the SSL certificate from drop down , if it’s not exist create one

   e. Keep the other settings to default

   Click on **Save changes**.

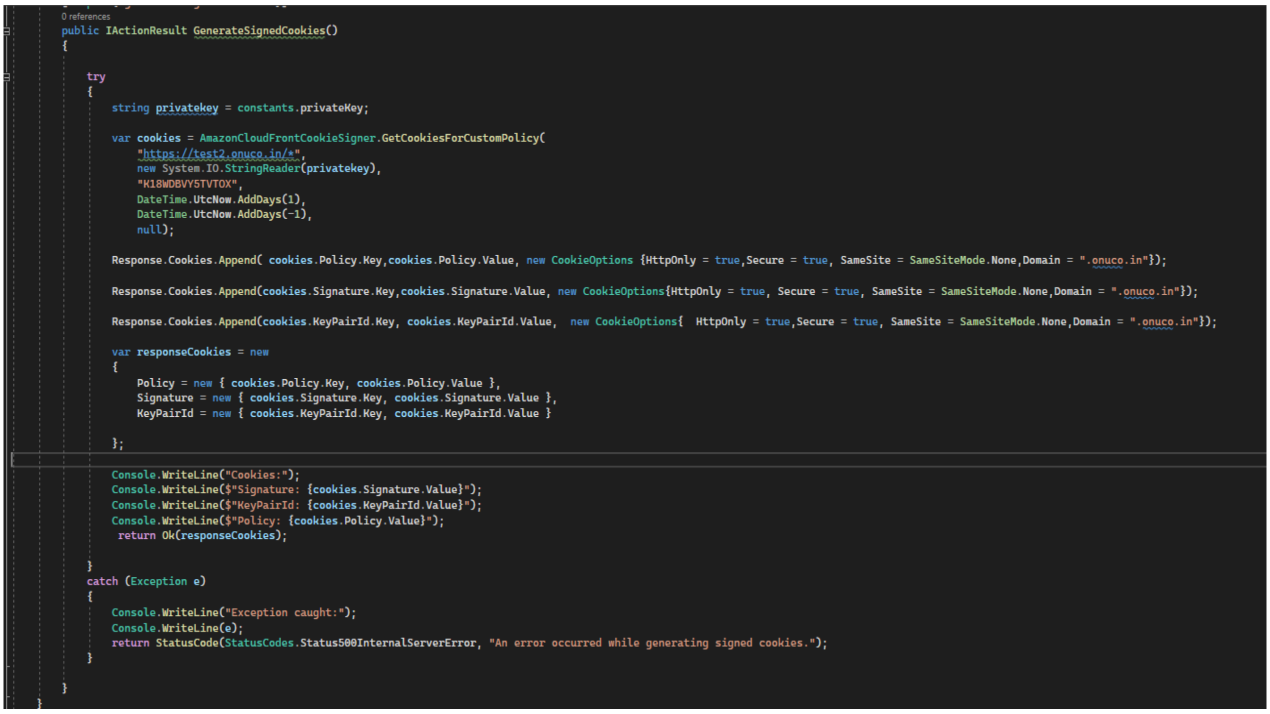
This will add Alternate domain name for cloudfront. Do the same for web app cloudfront distribution to add alternate domain name.

Step 3. Develop code for setting cookies using AmazonCloudFrontCookieSigner

Using AmazonCloudFrontCookiesigner create custom policy to set the signed cookies In ASP.net C#

The cookies contain a signature that CloudFront validates using a public-private key pair, allowing or denying access to the requested content based on the policy.

It requires distribution domain name , Private key , Key-Pair-Id, Expiration time, IP address(Optional).



```csharp
0 references
public IActionResult GenerateSignedCookies()
{

    try
    {
        string privatekey = constants.privateKey;

        var cookies = AmazonCloudFrontCookieSigner.GetCookiesForCustomPolicy(
            "https://test2.onuco.in/*",
            new System.IO.StringReader(privatekey),
            "K18WDBVY5TVTOX",
            DateTime.UtcNow.AddDays(1),
            DateTime.UtcNow.AddDays(-1),
            null);

        Response.Cookies.Append( cookies.Policy.Key,cookies.Policy.Value, new CookieOptions {HttpOnly = true,Secure = true, SameSite = SameSiteMode.None,Domain = ".onuco.in"});

        Response.Cookies.Append(cookies.Signature.Key,cookies.Signature.Value, new CookieOptions{HttpOnly = true, Secure = true, SameSite = SameSiteMode.None,Domain = ".onuco.in"});

        Response.Cookies.Append(cookies.KeyPairId.Key, cookies.KeyPairId.Value,  new CookieOptions{  HttpOnly = true,Secure = true, SameSite = SameSiteMode.None,Domain = ".onuco.in"});

        var responseCookies = new
        {
            Policy = new { cookies.Policy.Key, cookies.Policy.Value },
            Signature = new { cookies.Signature.Key, cookies.Signature.Value },
            KeyPairId = new { cookies.KeyPairId.Key, cookies.KeyPairId.Value }

        };

        Console.WriteLine("Cookies:");
        Console.WriteLine($"Signature: {cookies.Signature.Value}");
        Console.WriteLine($"KeyPairId: {cookies.KeyPairId.Value}");
        Console.WriteLine($"Policy: {cookies.Policy.Value}");
         return Ok(responseCookies);

    }
    catch (Exception e)
    {
        Console.WriteLine("Exception caught:");
        Console.WriteLine(e);
        return StatusCode(StatusCodes.Status500InternalServerError, "An error occurred while generating signed cookies.");
    }

}
}
```

- Distribution domain name – Cloudfront distribution domain
- Private Key – Private Key generated using OpenSSL
- Key-Pair-Id – ID of public key which is in cloudfront

Step 4. Link Custom domain to App Runner

Link Custom domain for App Runner

1. Sign in to the AWS Management Console and open the **AWS App Runner**
2. In the navigation menu, choose **Services**
3. Choose your Service
4. Choose **Custom Domains** tab under Service Overview
5. Click on **Link Domain**

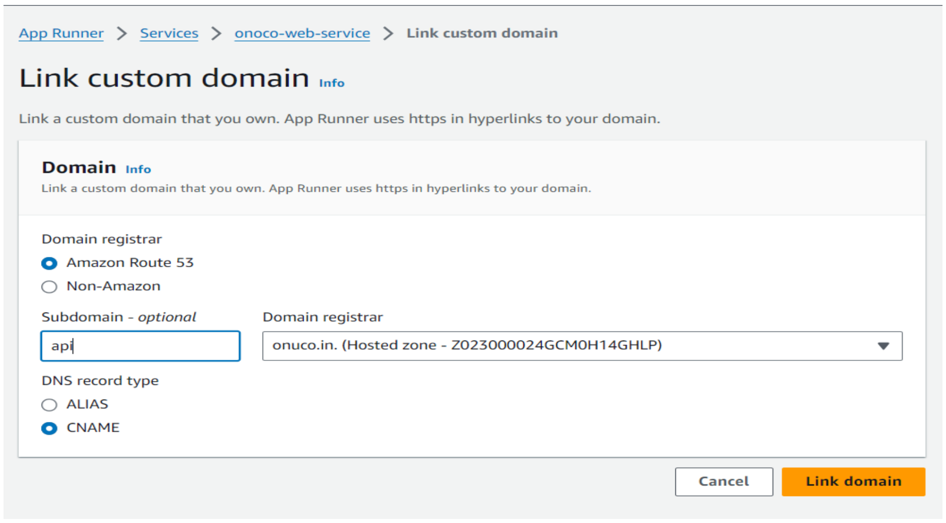
a. Select the Domain registrar i.e., Amazon Route 53

b. Subdomain, enter the subdomain (e.g., ‘api’)

c. Domain registrar , select your domain registrar from drop down (e.g., ‘onuco.in’)

d. DNS record type , select CNAME

When finished, Click on Link domain. It will take few minutes to link the domain.



App Runner > Services > onoco-web-service > Link custom domain

Link custom domain Info

Link a custom domain that you own. App Runner uses https in hyperlinks to your domain.

Domain Info

Link a custom domain that you own. App Runner uses https in hyperlinks to your domain.

Domain registrar
● Amazon Route 53
○ Non-Amazon

Subdomain - *optional*: api

Domain registrar: onuco.in. (Hosted zone - Z023000024GCM0H14GHLP)

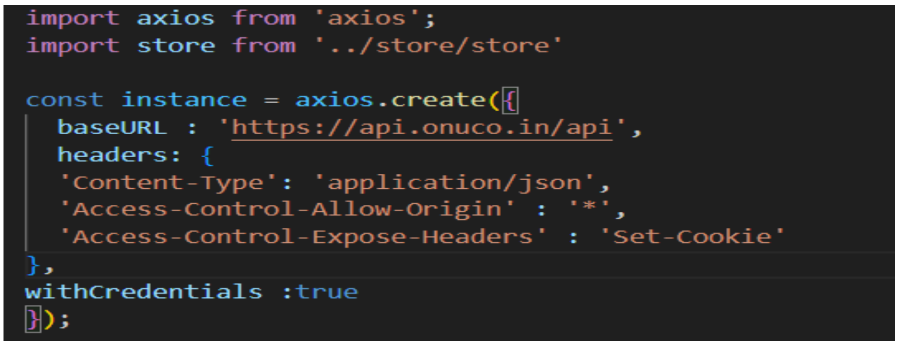
DNS record type
○ ALIAS
● CNAME

(Note: After linking the custom domain Configure DNS page will pop up , close it. It will automatically create custom domain in route 53)

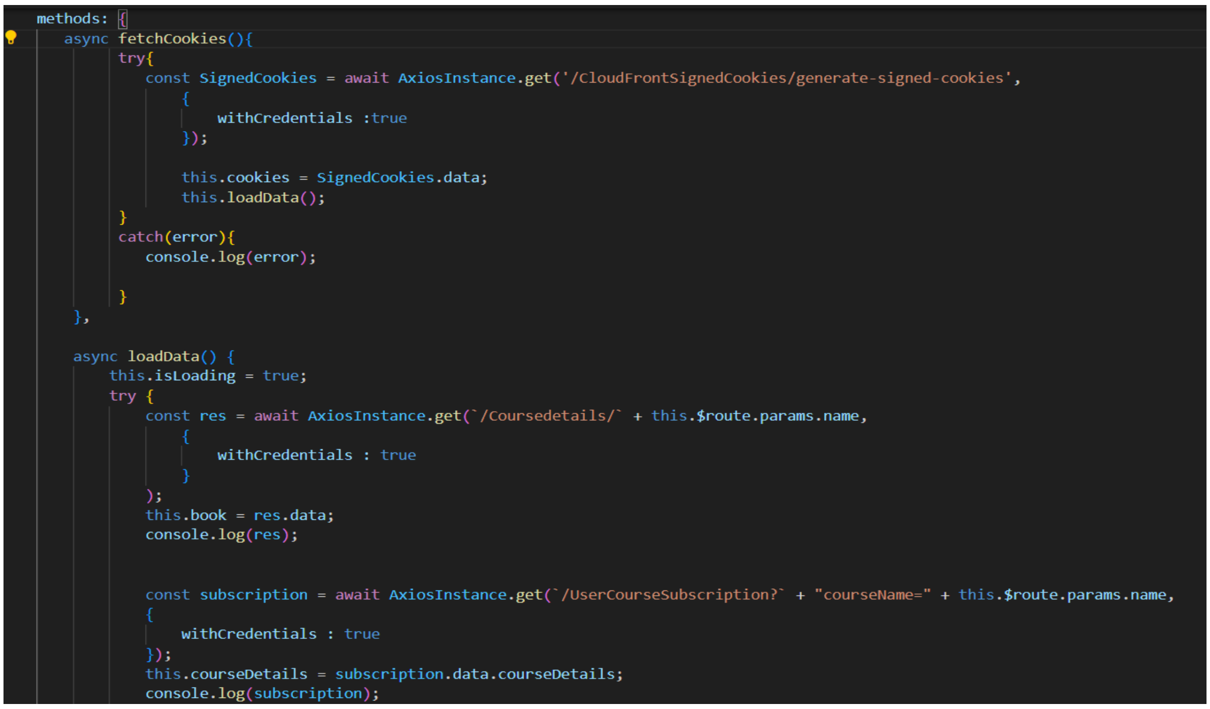
[Note: Keep all the resources(i.e., CloudFront, AppRunner,S3) under same domain(e.g., ‘ .onuco.in’)

Step 5. Using Set-Cookies in frontend (Vue.js)

Set **WithCredentials : true** for API endpoint to set the cookies, and set the headers.

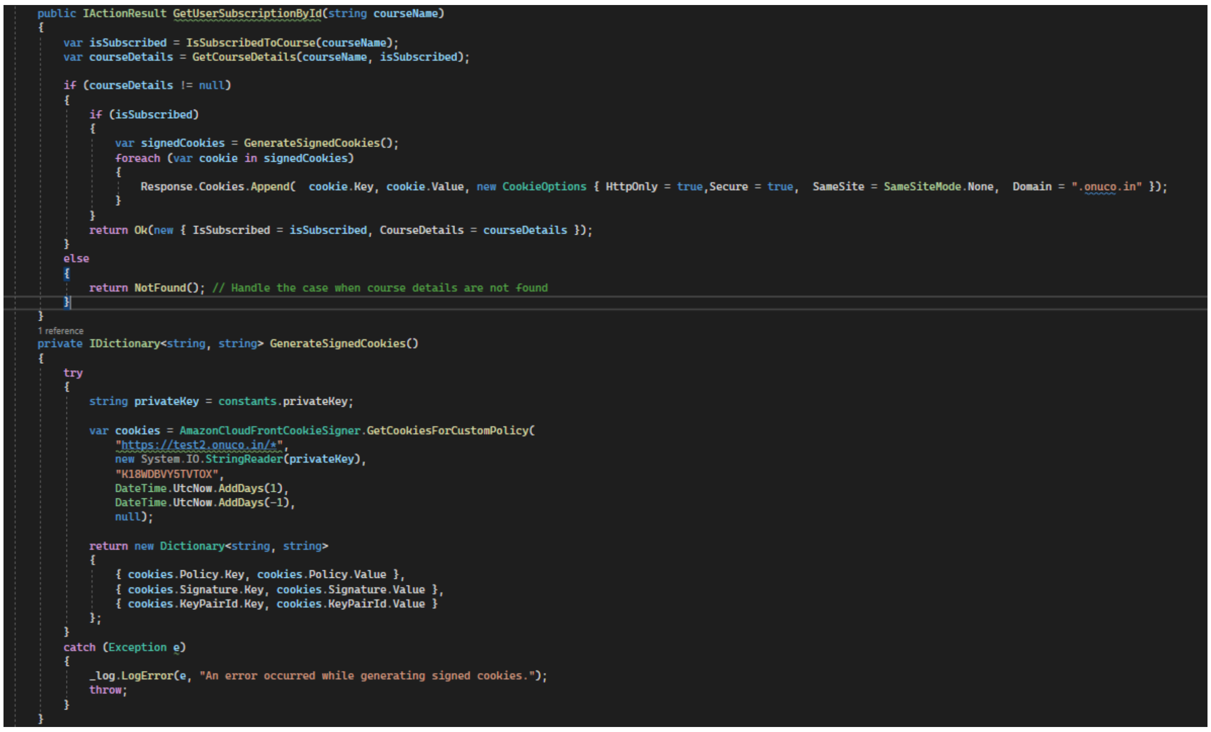


```
import axios from 'axios';
import store from '../store/store'

const instance = axios.create({
  baseURL : 'https://api.onuco.in/api',
  headers: {
  'Content-Type': 'application/json',
  'Access-Control-Allow-Origin' : '*',
  'Access-Control-Expose-Headers' : 'Set-Cookie'
},
withCredentials :true
});
```



```
methods: {
    async fetchCookies(){
        try{
            const SignedCookies = await AxiosInstance.get('/CloudFrontSignedCookies/generate-signed-cookies',
                {
                    withCredentials :true
                });

                this.cookies = SignedCookies.data;
                this.loadData();
        }
        catch(error){
            console.log(error);

        }
    },

    async loadData() {
        this.isLoading = true;
        try {
            const res = await AxiosInstance.get(`/Coursedetails/` + this.$route.params.name,
                {
                    withCredentials : true
                }
            );
            this.book = res.data;
            console.log(res);


            const subscription = await AxiosInstance.get(`/UserCourseSubscription?` + "courseName=" + this.$route.params.name,
            {
                withCredentials : true
            });
            this.courseDetails = subscription.data.courseDetails;
            console.log(subscription);
```

Alternate backend code for checking user subscription and set cookies for subscribed users



```
public IActionResult GetUserSubscriptionById(string courseName)
{
    var isSubscribed = IsSubscribedToCourse(courseName);
    var courseDetails = GetCourseDetails(courseName, isSubscribed);

    if (courseDetails != null)
    {
        if (isSubscribed)
        {
            var signedCookies = GenerateSignedCookies();
            foreach (var cookie in signedCookies)
            {
                Response.Cookies.Append(  cookie.Key, cookie.Value, new CookieOptions { HttpOnly = true,Secure = true,  SameSite = SameSiteMode.None,  Domain = ".onuco.in" });
            }
        }
        return Ok(new { IsSubscribed = isSubscribed, CourseDetails = courseDetails });
    }
    else
    {
        return NotFound(); // Handle the case when course details are not found
    }
}
1 reference
private IDictionary<string, string> GenerateSignedCookies()
{
    try
    {
        string privateKey = constants.privateKey;

        var cookies = AmazonCloudFrontCookieSigner.GetCookiesForCustomPolicy(
            "https://test2.onuco.in/*",
            new System.IO.StringReader(privateKey),
            "K18WDBVY5TVTOX",
            DateTime.UtcNow.AddDays(1),
            DateTime.UtcNow.AddDays(-1),
            null);

        return new Dictionary<string, string>
        {
            { cookies.Policy.Key, cookies.Policy.Value },
            { cookies.Signature.Key, cookies.Signature.Value },
            { cookies.KeyPairId.Key, cookies.KeyPairId.Value }
        };
    }
    catch (Exception e)
    {
        _log.LogError(e, "An error occurred while generating signed cookies.");
        throw;
    }
}
```