😊 Lambda@edge and CloudFront Distribution

Lambda@Edge is an extension of AWS Lambda, designed to allow developers to run code closer to the end users of their applications, improving performance and reducing latency. It integrates with Amazon CloudFront, which is AWS's content delivery network (CDN), enabling you to execute code in AWS data centers located globally, called edge locations.

Key Features of Lambda@Edge:

- **Low Latency**: By executing code at CloudFront edge locations, it reduces the time it takes for a request to travel back to the origin server, enhancing response times for users around the world.
- **Event-Driven**: Lambda@Edge is triggered by specific CloudFront events, such as viewer requests, origin requests, viewer responses, and origin responses.
- **Serverless**: As with AWS Lambda, you don't need to manage servers. AWS takes care of scaling and infrastructure management.
- **Scalability**: Automatically scales based on the number of requests, handling any traffic load.
- **Security**: You can use Lambda@Edge to inspect and modify headers, authenticate users, and enforce authorization logic right at the edge.

Common Use Cases:

- **A/B Testing**: Serve different versions of your website to different users without deploying multiple versions of your site.
- **User Authentication and Authorization**: Authenticate users at the edge before requests reach your origin server.
- **Dynamic Content Manipulation**: Modify HTTP headers or generate HTTP responses dynamically based on request parameters.
- **SEO Optimization**: Modify URLs and redirect traffic based on user location or device type to optimize search engine rankings.

Lambda@Edge is particularly useful for applications that require quick, dynamic responses and need to be globally distributed to users.

A **CloudFront distribution** is a configuration setup within Amazon CloudFront, AWS's content delivery network (CDN) service. It defines how CloudFront should deliver your content (like web pages, images, videos, and other assets) to users across the globe.

Key Components of a CloudFront Distribution:

- **Origin**: The source of the content that CloudFront will distribute. This can be an Amazon S3 bucket, an HTTP server (like an EC2 instance), or an external web server.
- **Edge Locations**: These are data centers located around the world where CloudFront caches copies of your content to deliver it to users with low latency.
- **Cache Behavior**: Defines how CloudFront handles requests for content. You can set up multiple cache behaviors for different parts of your site or app, specifying the origin, caching policy, and response headers for each.
- **Distribution Types**:
    - **Web Distribution**: Used for serving static and dynamic content over HTTP or HTTPS, like websites, APIs, or media files.
    - **RTMP Distribution**: (Deprecated) Used for streaming media files using Adobe Flash Media Server's RTMP protocol.
- **Domain Name**: CloudFront assigns a unique domain name to your distribution, which you can use in your URLs or configure your custom domain to point to it.
- **SSL/TLS Configuration**: You can secure your content by using HTTPS and integrating your own SSL/TLS certificates.
- **Access Control**: CloudFront offers various security features, like signed URLs, signed cookies, and geoblocking, to restrict access to your content.

How CloudFront Distribution Works:

1. **User Request**: A user requests content through a URL pointing to your CloudFront distribution.
2. **Edge Location Check**: CloudFront checks if the requested content is already cached at the nearest edge location.
3. **Cache Hit**: If the content is cached (a cache hit), CloudFront serves it directly to the user, reducing latency.
4. **Cache Miss**: If the content is not cached (a cache miss), CloudFront fetches it from the origin, caches it at the edge location, and then serves it to the user.
5. **Content Delivery**: Subsequent requests for the same content will be served from the cache, improving delivery speed.

Benefits of Using CloudFront Distributions:

- **Global Reach**: Deliver content to users from the nearest edge location, reducing latency.
- **Scalability**: Automatically handles varying levels of traffic, from low to very high.
- **Security**: Integrated with AWS Shield, AWS WAF, and SSL/TLS to provide robust security for your content.

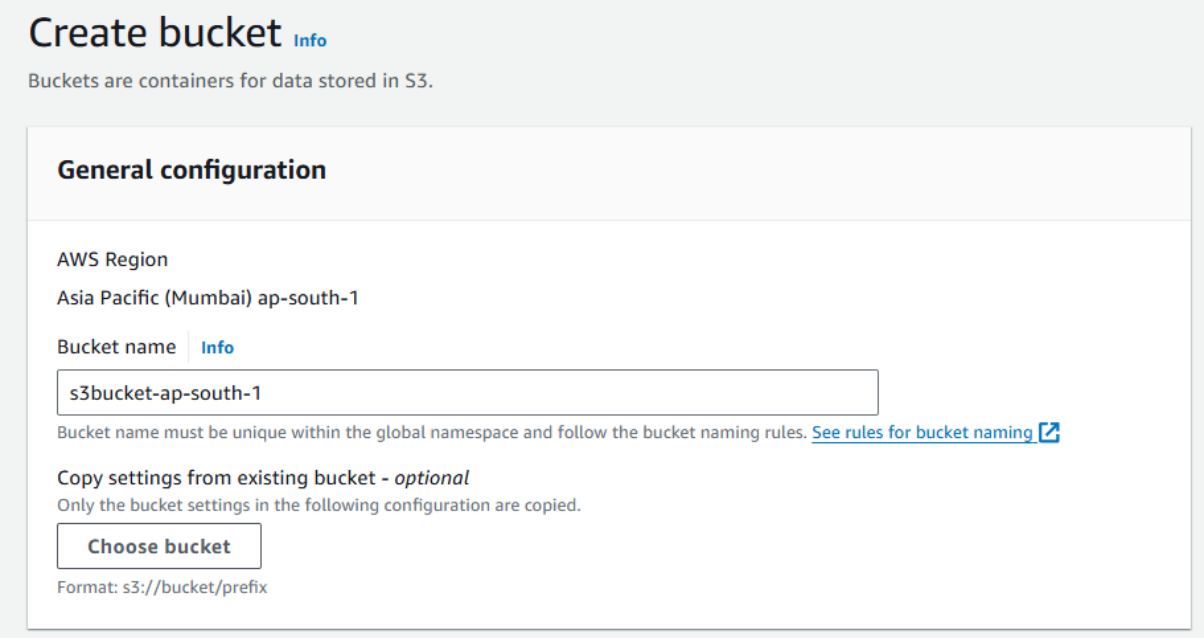
- **Cost Efficiency**: Reduces the load on your origin servers, potentially lowering hosting costs.
- **Customizability**: Fine-tune cache behaviour, security settings, and more to meet specific needs.

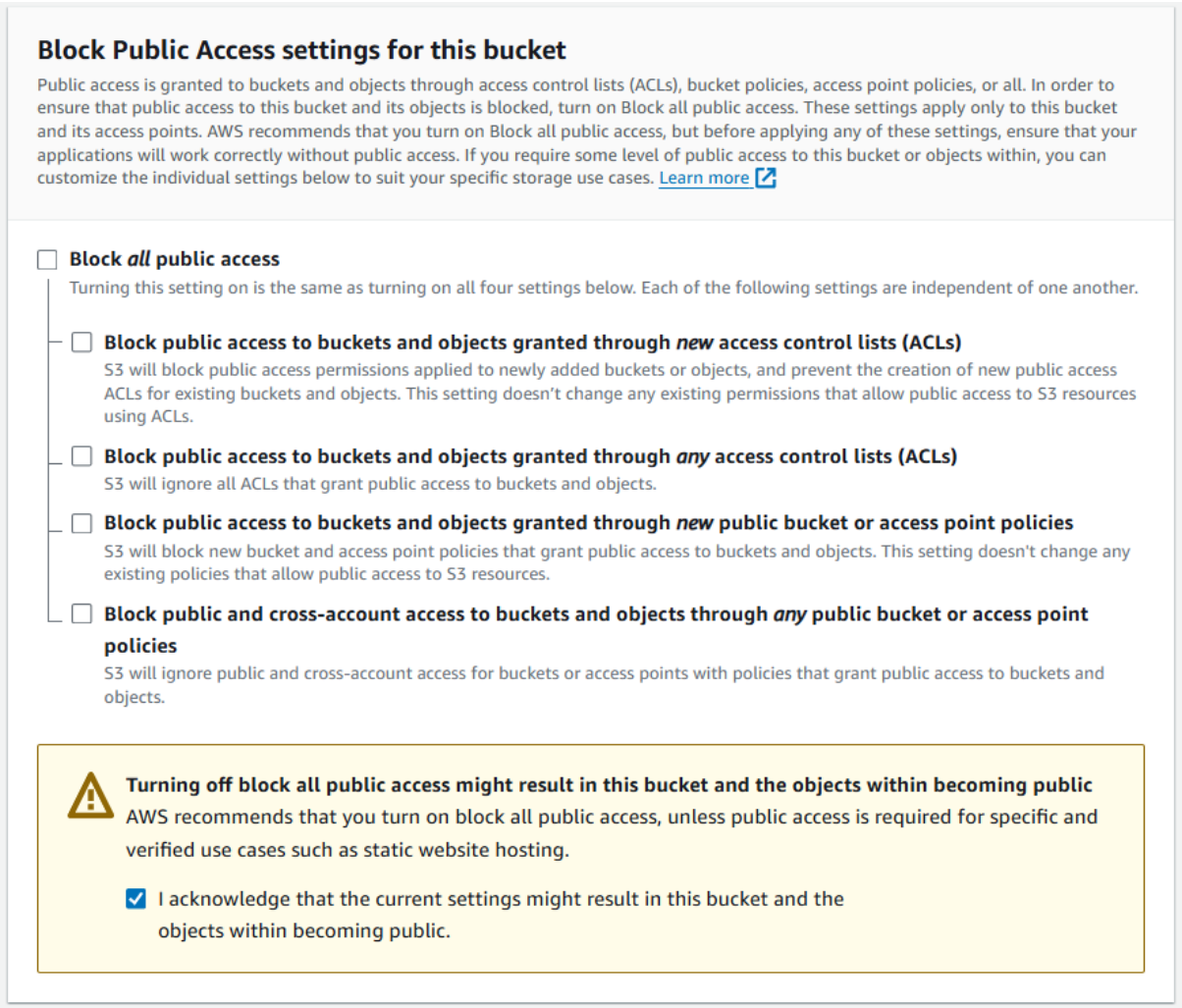
CloudFront distributions are essential for delivering content efficiently and securely to a global audience.

In this lab, you'll set up a static website using Amazon S3 and CloudFront. You'll create a public S3 bucket in Mumbai, upload your website, and configure it for hosting. Then, you'll set up CloudFront to distribute your site globally and use Lambda@Edge to direct visitors to the nearest server based on their location. The end goal is to have a globally accessible website that loads efficiently by automatically routing users to the closest server.

😄 To begin with the Lab:

1. In this lab we are going to deploy an Amazon S3 bucket in ap-south-1 which is Mumbai region.
2. Also, we will be hosting a static website on our bucket,
3. Now move to S3 in your console and create your bucket. Give it a name and scroll down. Give your bucket name in this format as shown below.
4. Then you should enable public access to your bucket and create it.

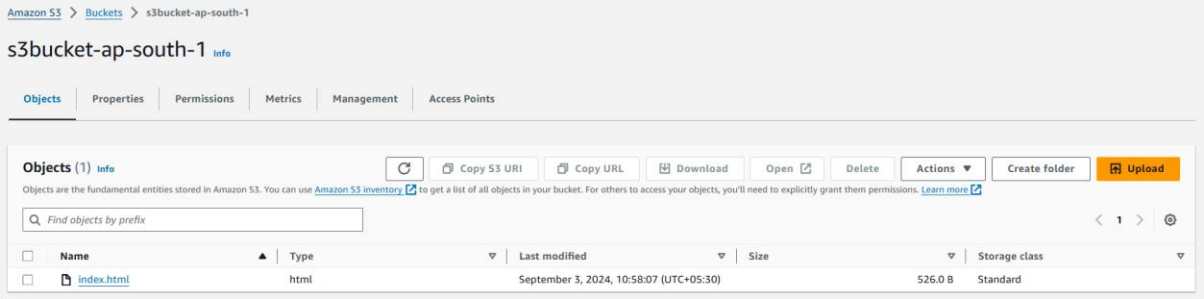


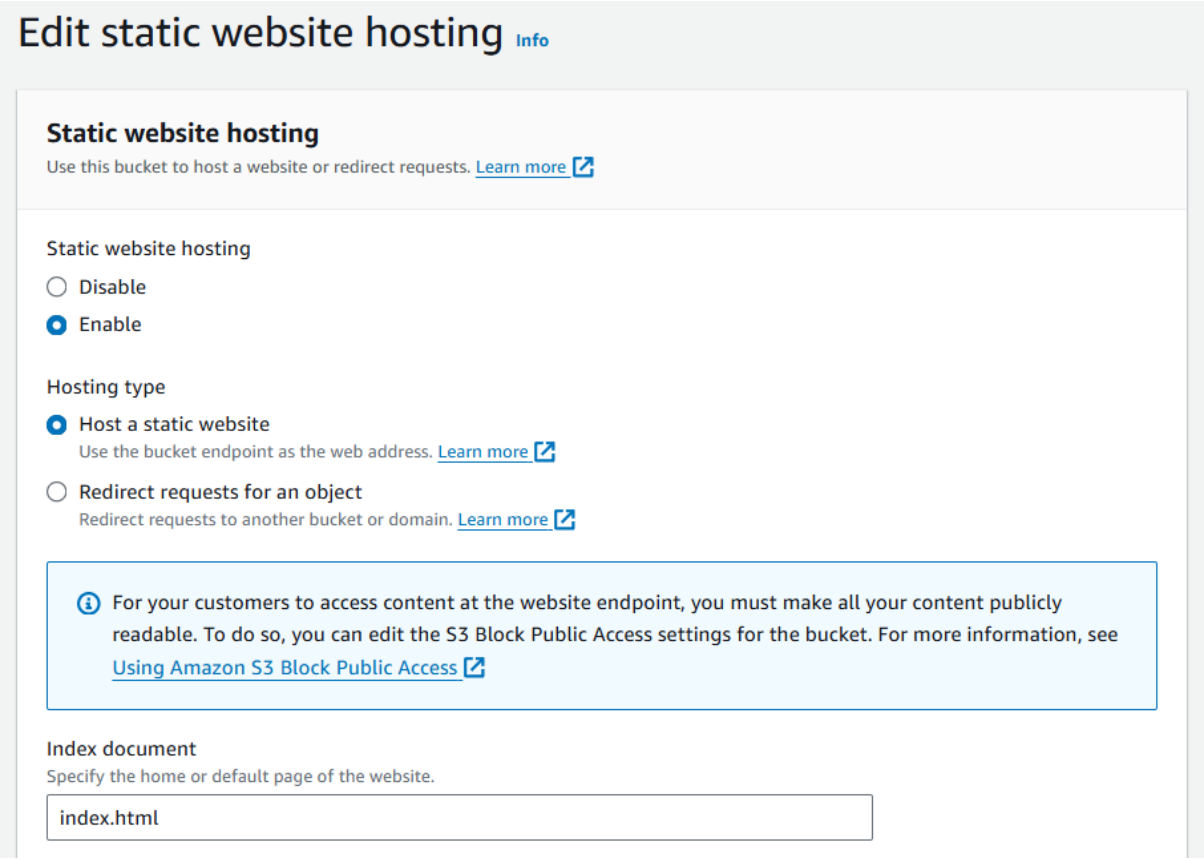


5. So, you have created your public bucket but we have not updated the policy yet, you are going to update the bucket policy now using the below code.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::s3bucket-ap-south-1/*"
    }
  ]
}
```

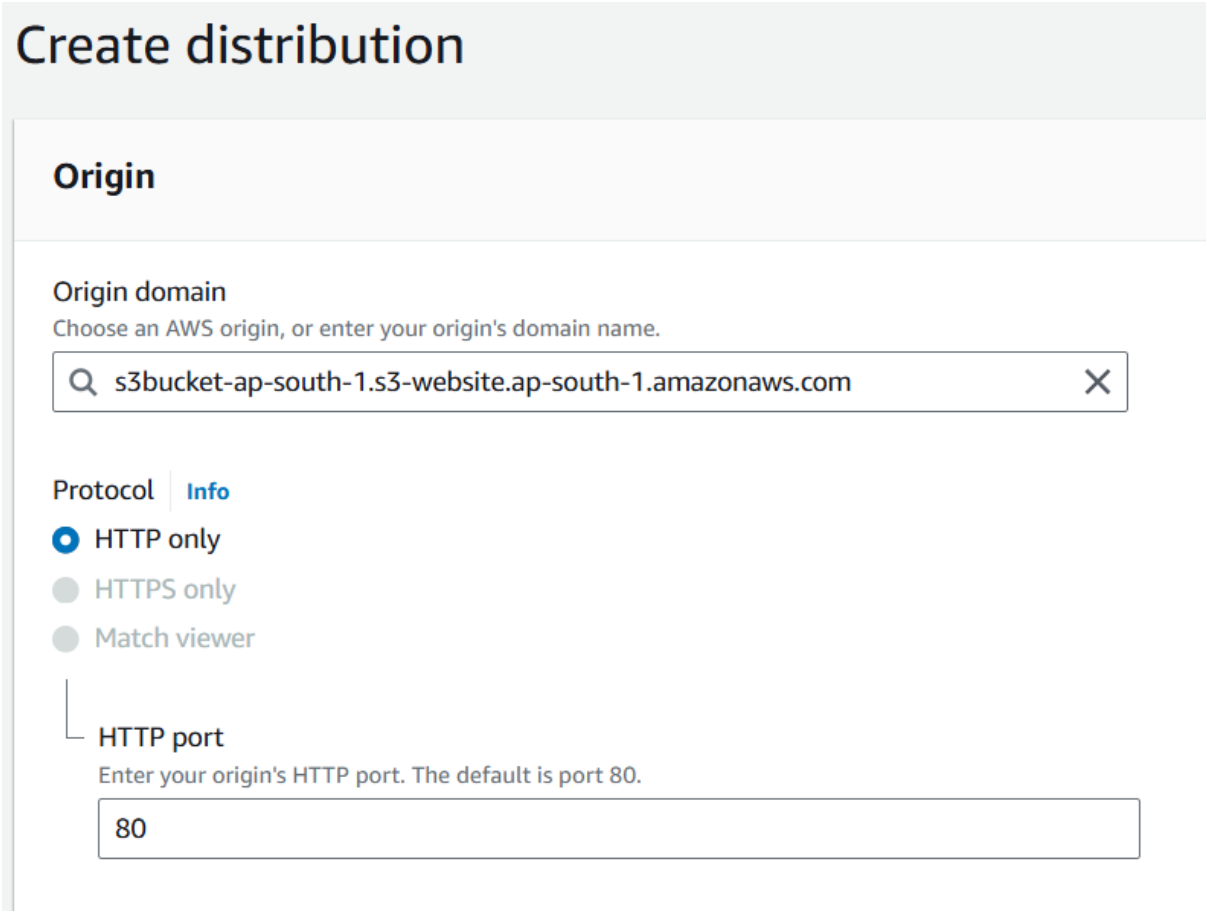
6. In our Mumbai bucket we have uploaded our static website and we are going to enable static website hosting.



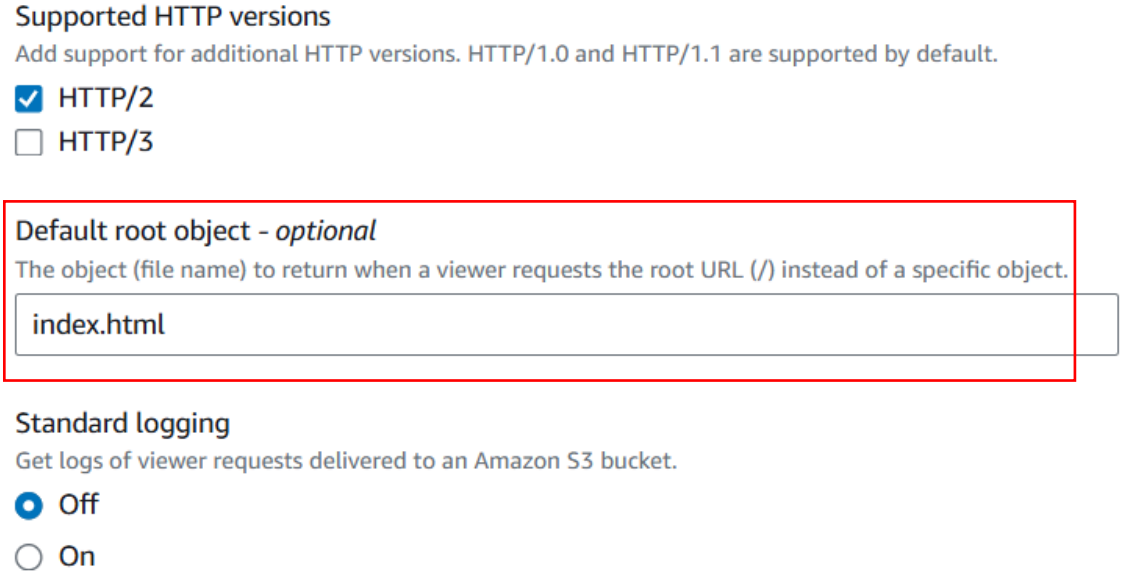


7. Below you can see our website has been hosted successfully. You can get this Index.html file from our GitHub.

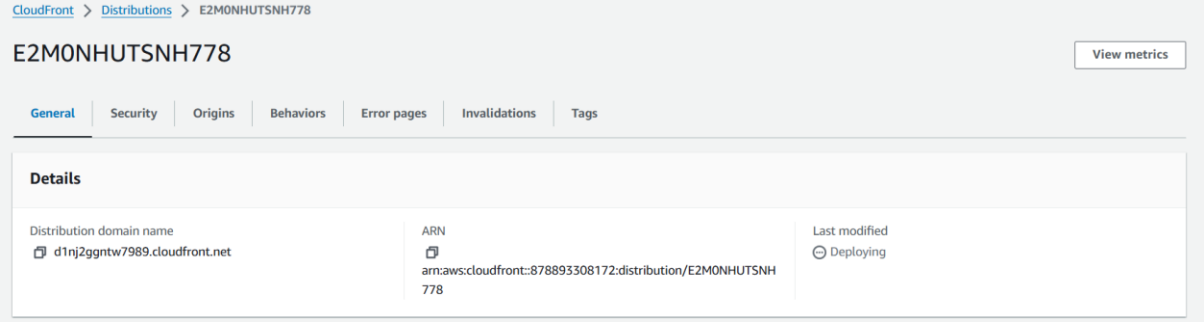
8. We have completed all the work for S3. Now we are going to CloudFront to create our distribution.
9. While creating your distribution you must choose the Mumbai bucket as your default bucket. Then scroll down to the bottom.



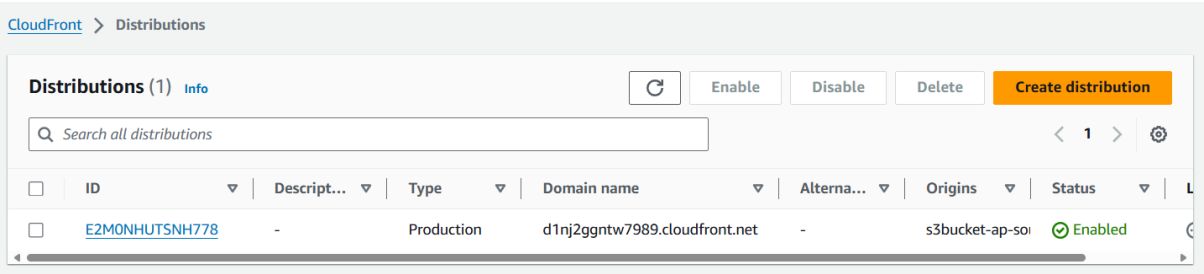
10. Here you will find the default root object option, you must write here index.html and create your distribution.



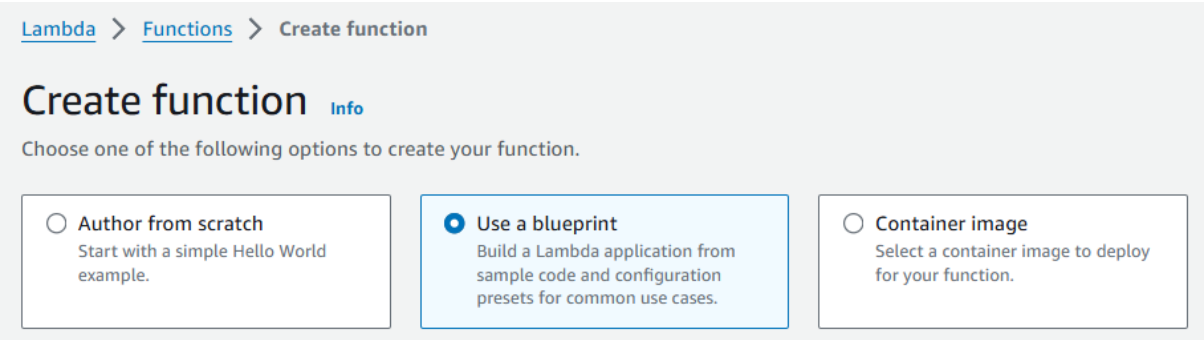
11. The deployment of your distribution might take some time.



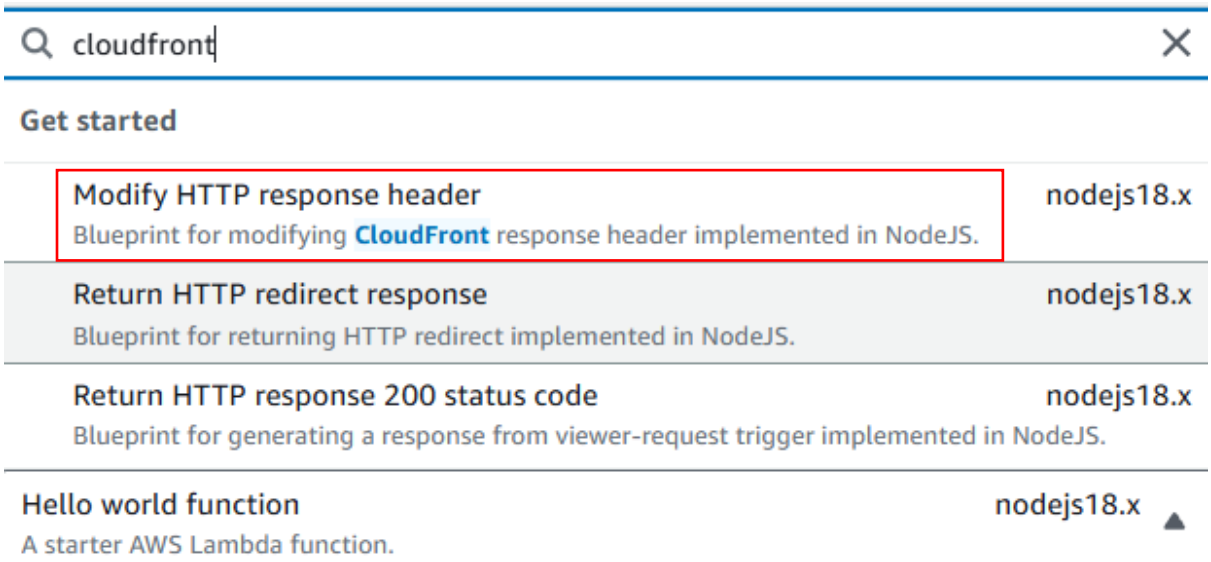
12. When the status of your CloudFront is enabled, you must copy the domain name and paste it into a new tab. Then you will see your default Mumbai region S3 bucket static hosted website here.



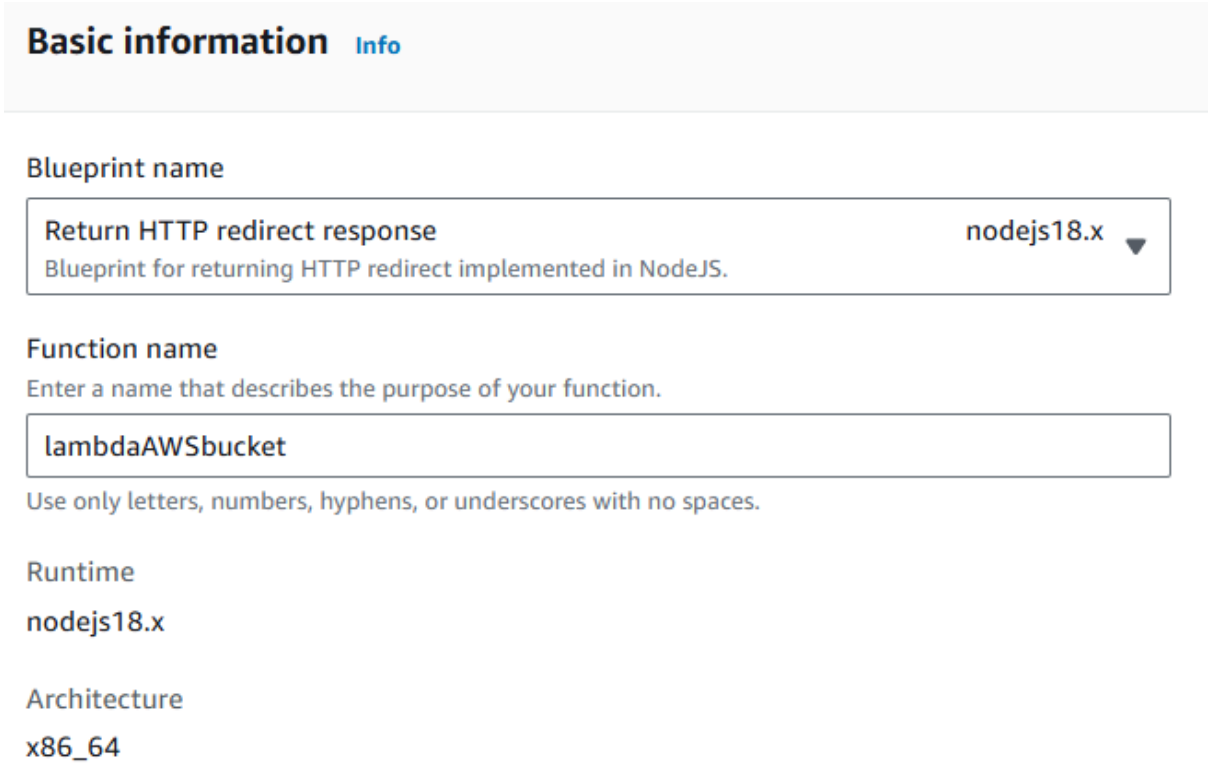
13. Now we must create a **lambda@edge function** that will redirect our traffic based on our origin. So, go to lambda and click on the create function. **Just remember to create your lambda function in N. Virginia Region.**
14. Here you must choose Use a blueprint and scroll down.



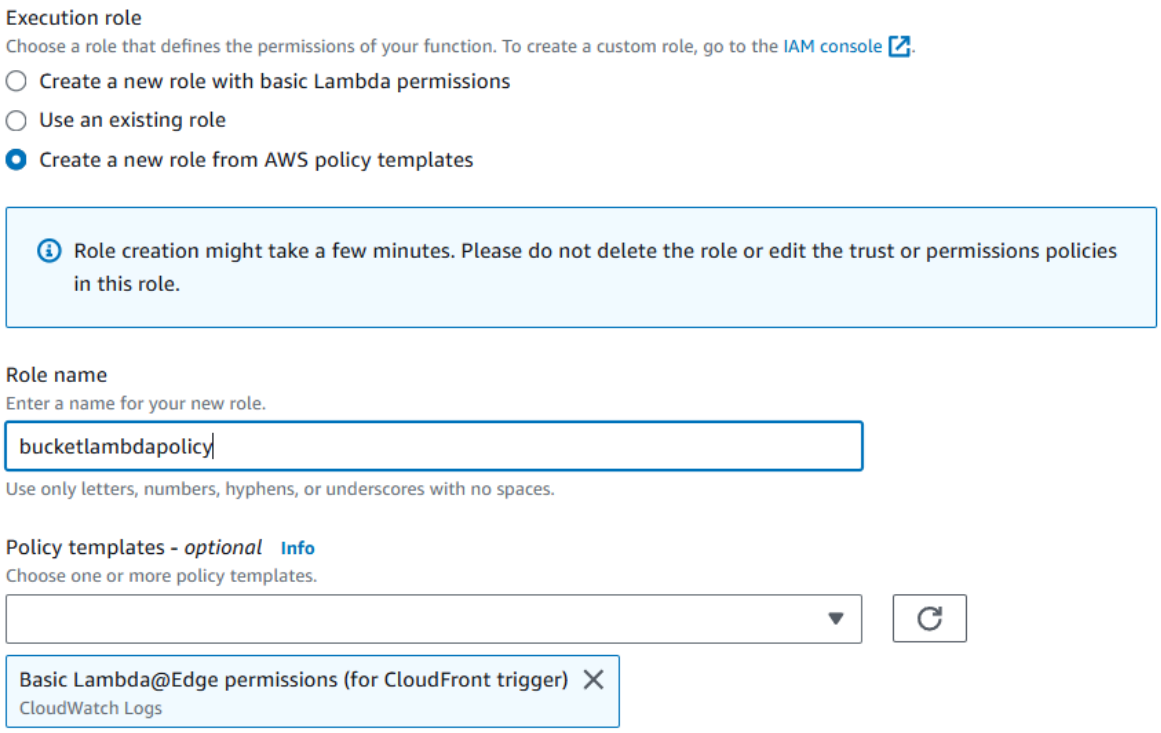
15. Then in the blueprint name search for CloudFront and choose Modify HTTP response header.



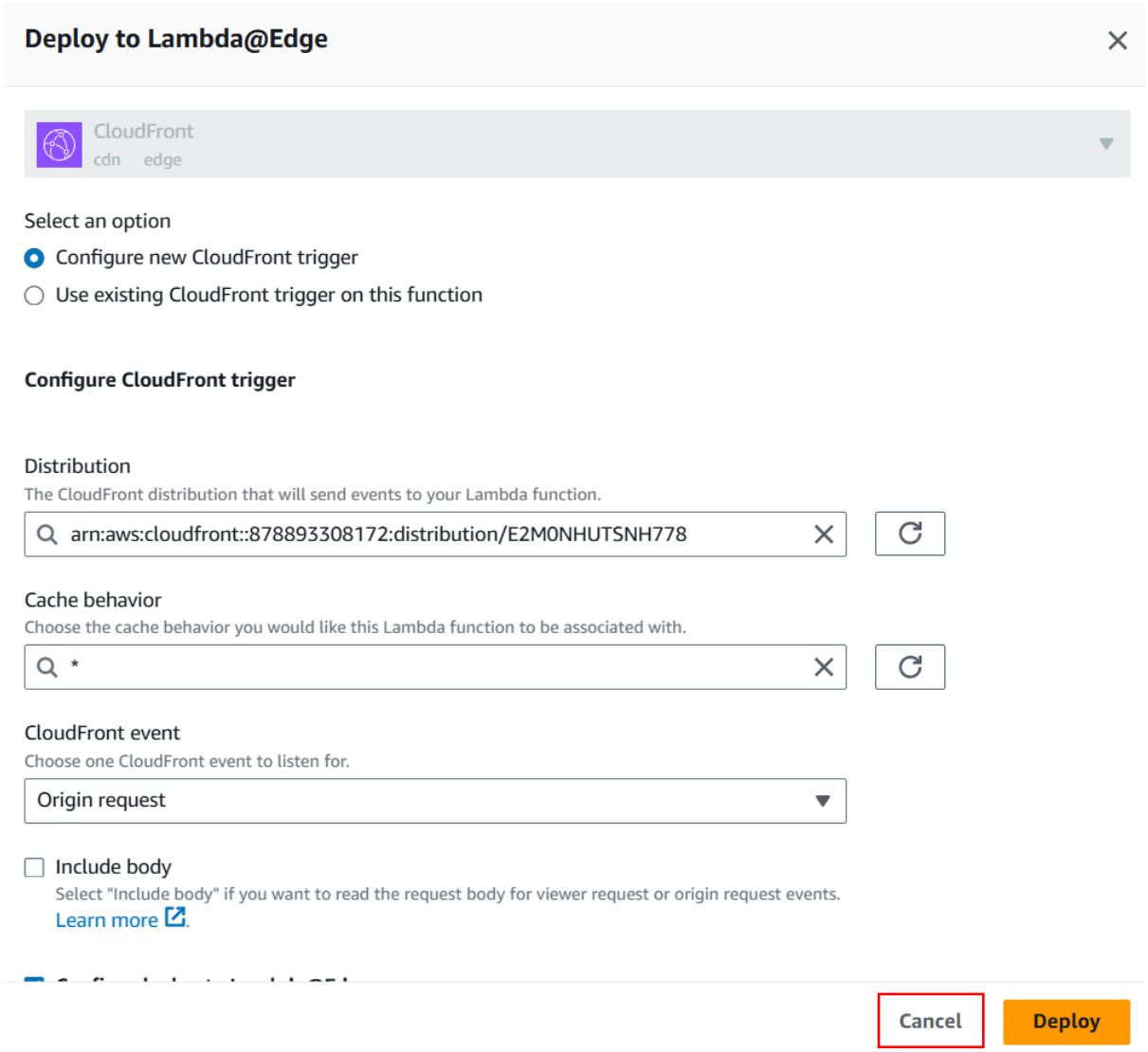
16. Now give your lambda function a name and scroll down.



17. For the execution role choose to create a new role, then give your role a name and scroll down to the bottom to create your lambda function.



18. Then you will see this type of pop-up window, **here you must choose to cancel this window**. We will configure our cloud front and deploy it to **Lambda@edge** later.



19. So, follow the below link for AWS documentation and scroll down to this example shown in the snapshot. Copy the code given in this example and paste it into your lambda function.
20. Remember to give your S3 bucket name in the code.

    https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/lambda-examples.html

Example: Use an origin-request trigger to change the Amazon S3 origin Region

This function demonstrates how an origin-request trigger can be used to change the Amazon S3 origin from which the content is fetched, based on request properties.

In this example, we use the value of the `CloudFront-Viewer-Country` header to update the S3 bucket domain name to a bucket in a Region that is closer to the viewer. This can be useful in several ways:

- It reduces latencies when the Region specified is nearer to the viewer's country.
- It provides data sovereignty by making sure that data is served from an origin that's in the same country that the request came from.

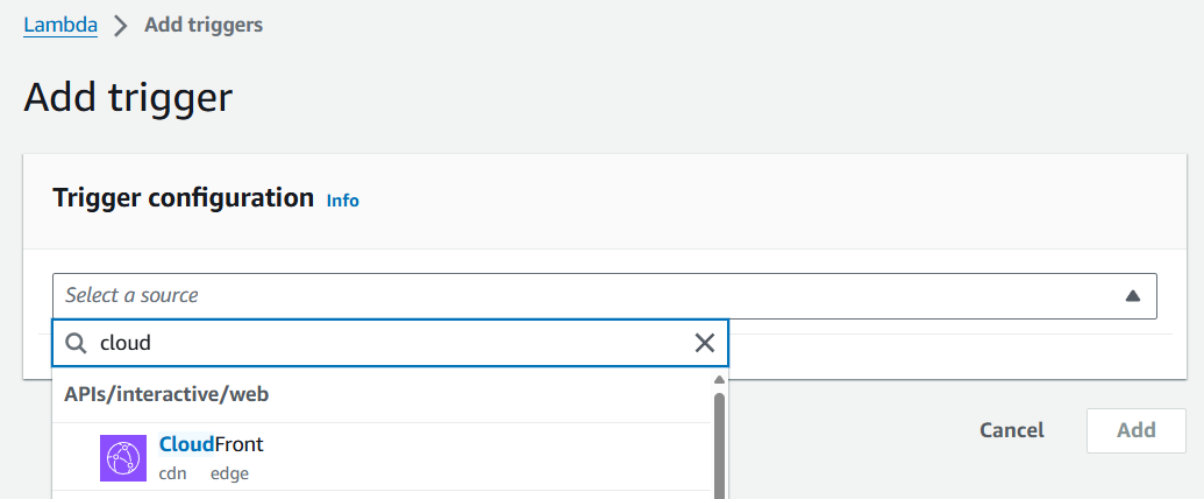
To use this example, you must do the following:

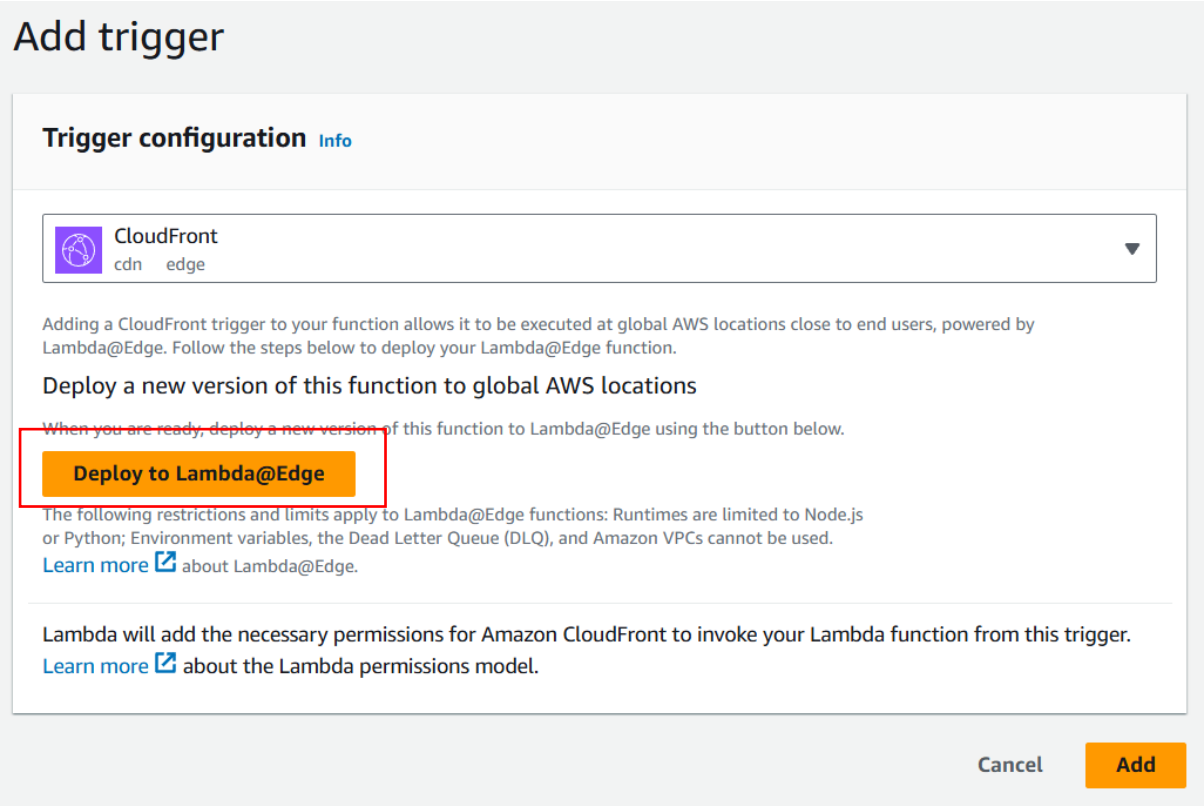
- Configure your distribution to cache based on the `CloudFront-Viewer-Country` header. For more information, see Cache based on selected request headers.
- Create a trigger for this function in the origin request event. CloudFront adds the `CloudFront-Viewer-Country` header after the viewer request event, so to use this example, you must make sure that the function executes for an origin request.

```
'use strict';

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;

    const countryToRegion = {
        'SG': 'ap-southeast-1',  // Singapore
        'US': 'us-east-1',       // North Virginia
        'IN': 'ap-south-1'       // India
    };

    if (request.headers['cloudfront-viewer-country']) {
        const countryCode = request.headers['cloudfront-viewer-country'][0].value;
        const region = countryToRegion[countryCode];

        if (region) {
            request.origin.s3.region = region;
            const domainName = `s3bucket-${region}.s3.${region}.amazonaws.com`;
            request.origin.s3.domainName = domainName;
            request.headers['host'] = [{ key: 'host', value: domainName }];
        }
    }
```

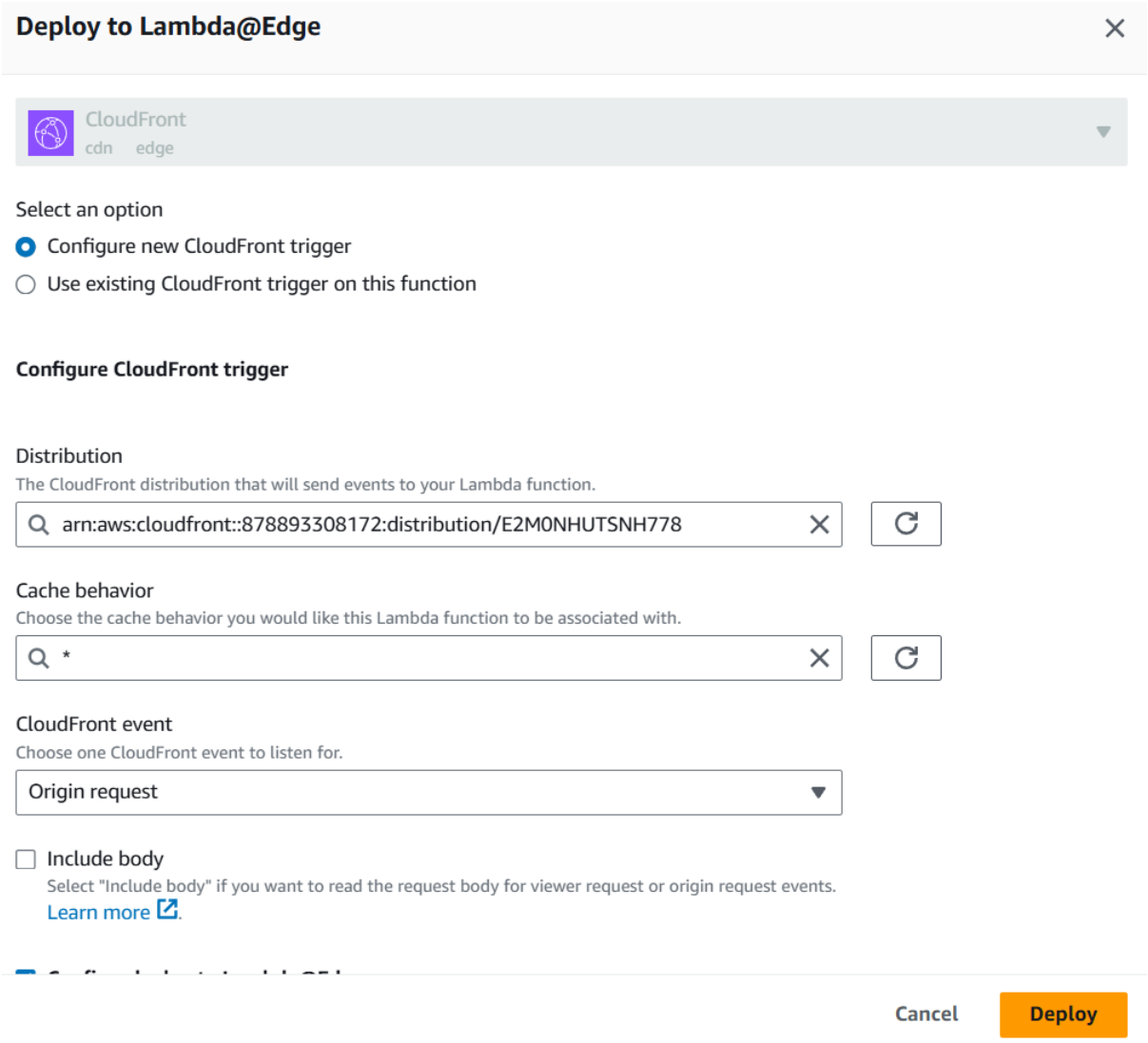
```
    callback(null, request);
};
```

21. Once you have changed the code then you need to click on deploy the code. After that, the code is deployed and you must click on add trigger in your lambda function.
22. Here you must choose CloudFront. Then click on deploy to Lambda@Edge.

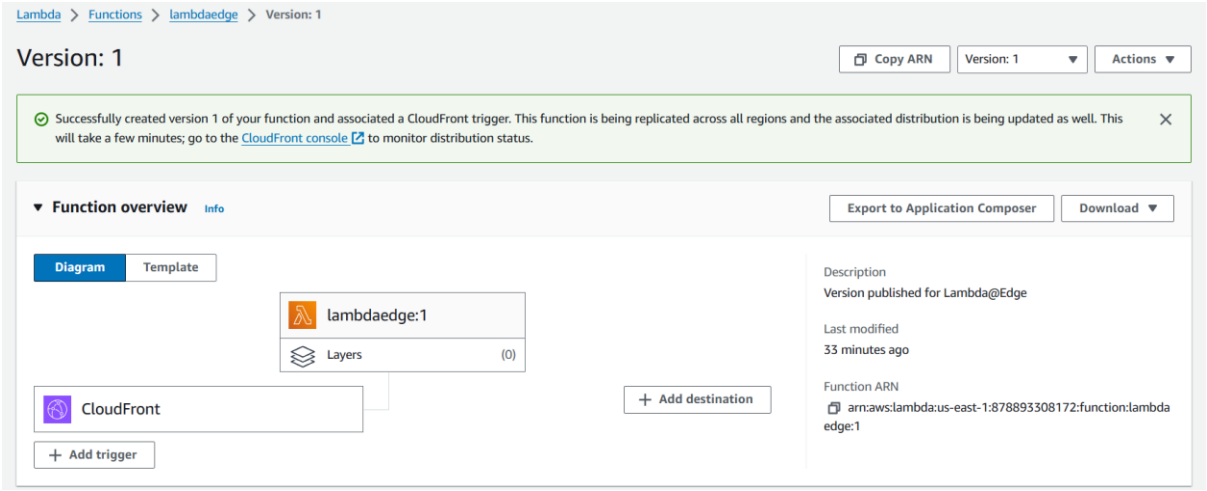




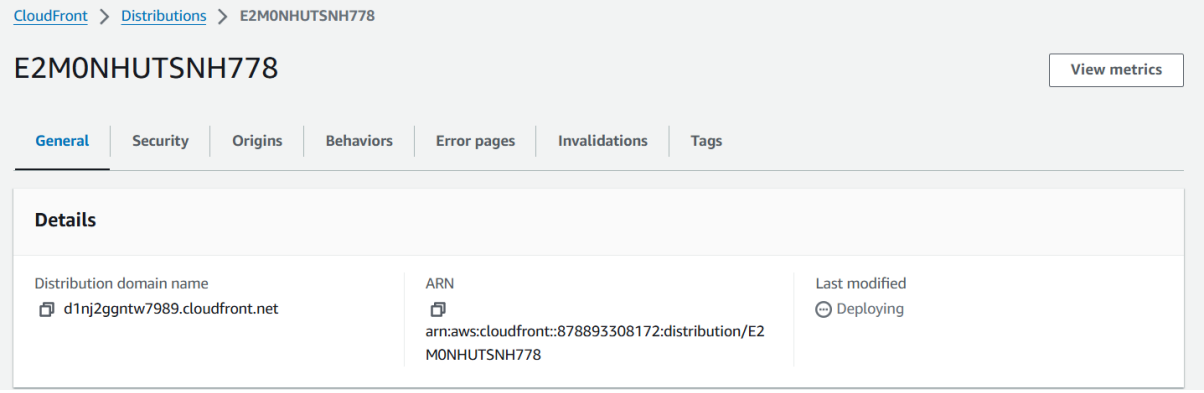
23. Below you can see that it is repopulating our CloudFront behaviour, so choose your distribution and click on deploy.



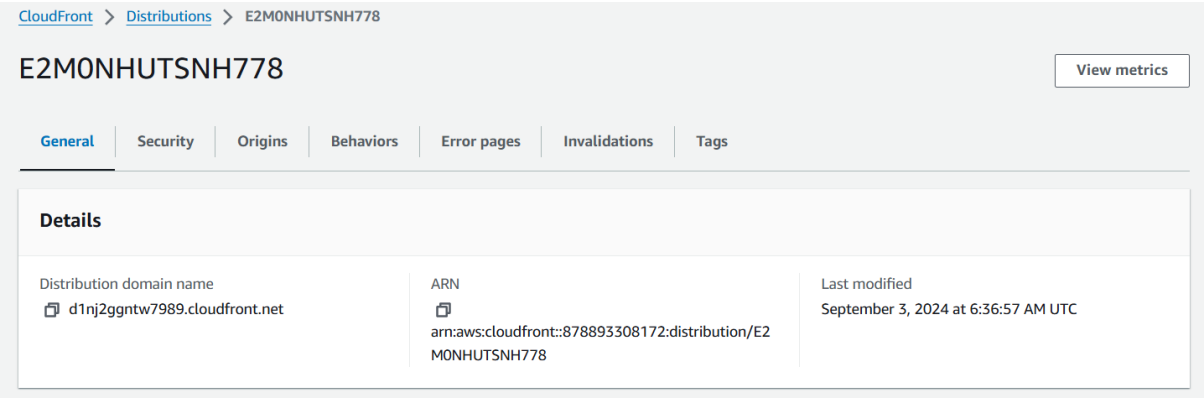
24. Below you can see that this is version 1 of our lambda function.



25. Now if you go to CloudFront you can see that it again started to deploy.



26. Now you must wait for some time until your distribution has been deployed. Once it is deployed, then copy the distribution domain name and paste it into a new tab.



27. Below you can see that using the domain name we can see our website, hosted in our Mumbai S3 bucket.

Your website is hosted in:

ap-south-1 (DefaultRegionMumbai)