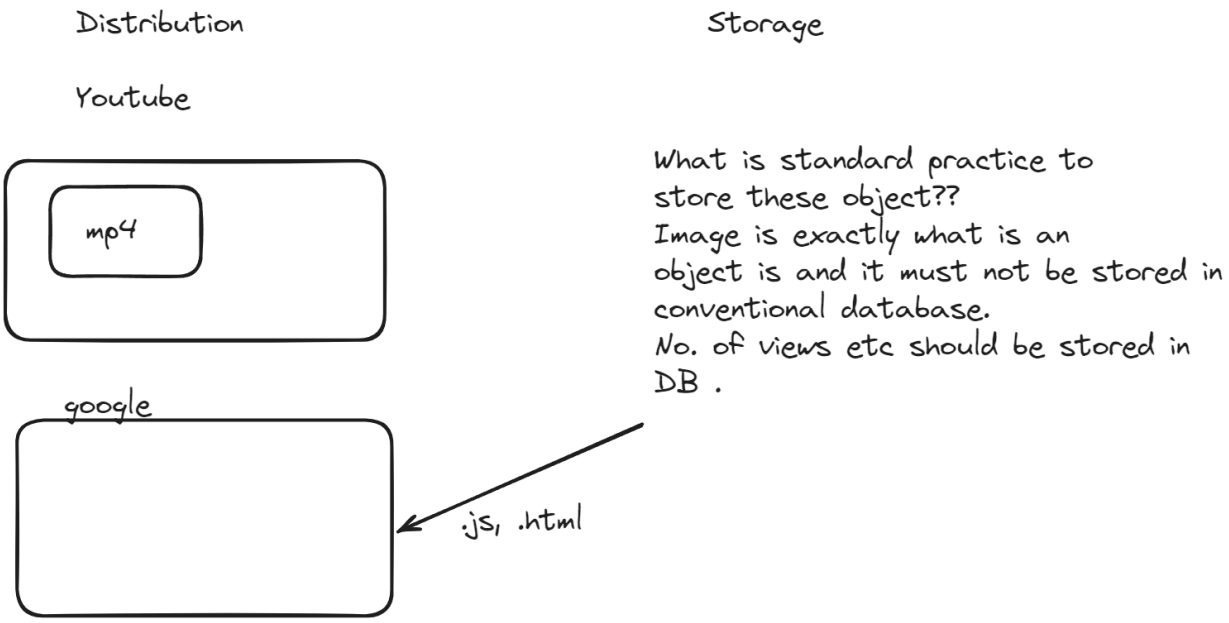
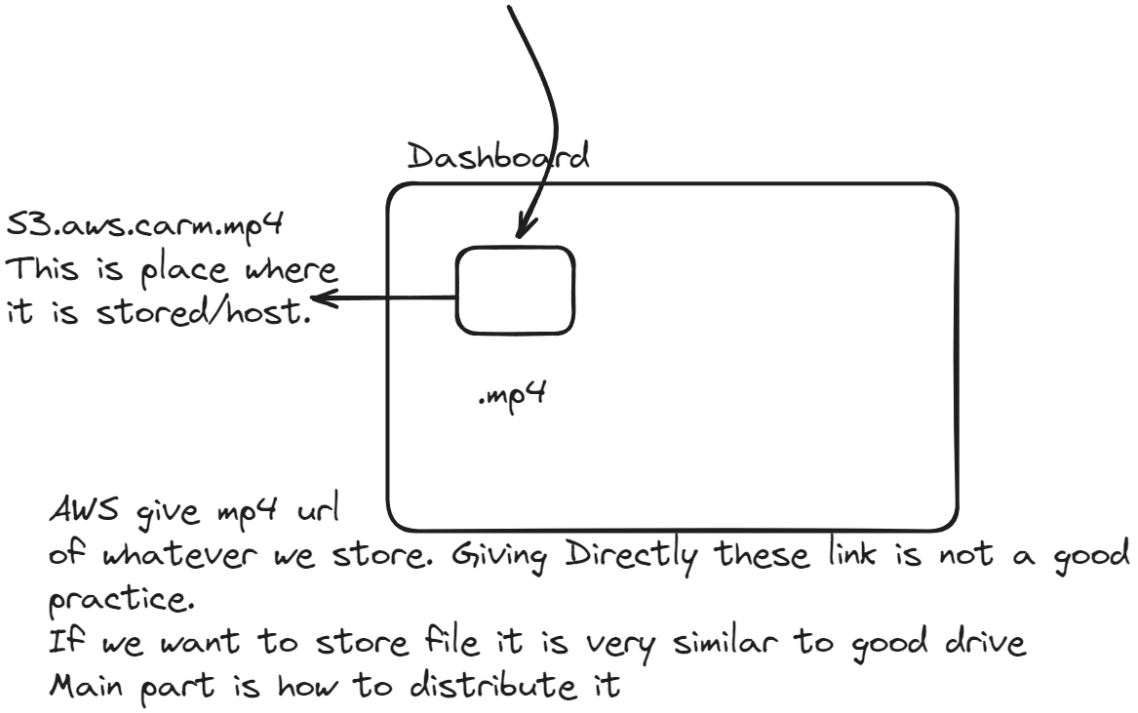
Deploying Frontend on AWS

(https://projects.100xdevs.com/tracks/w5E6PT2t0IyOFM3bZxcM/aws-fe-1)



Object stores where we store these objects, every cloud provide have it for example in case of AWS it is S3 **(simple storage service)** .



We need to know about **CDNs** .

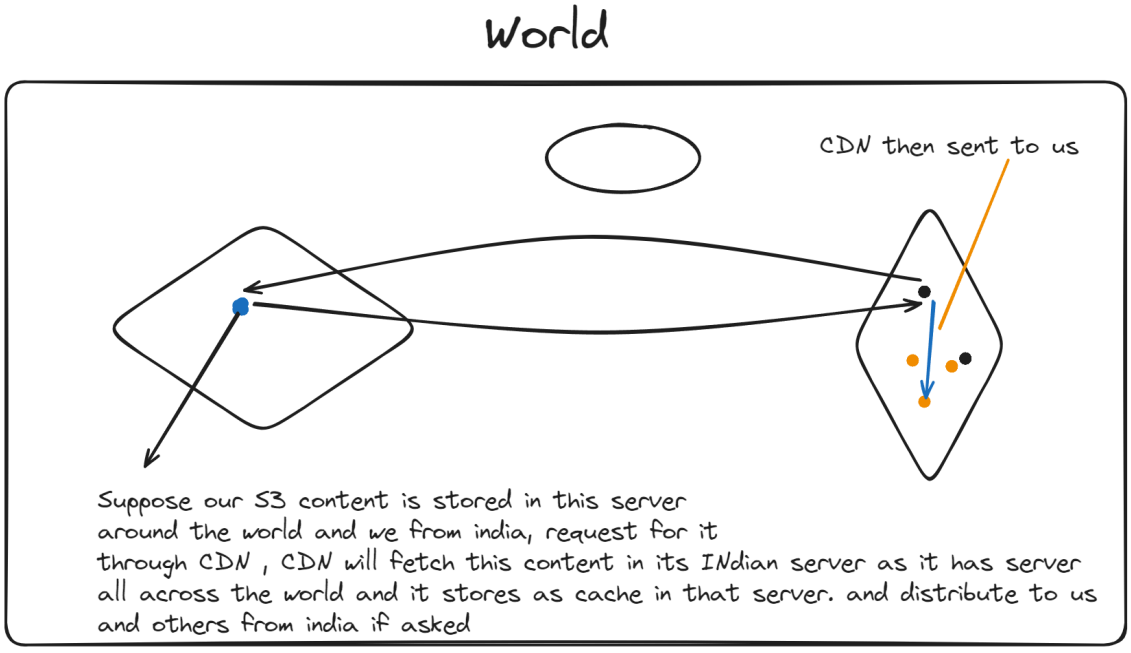
Storage — **object stores**

Distribution —- **CDNs (Content Delivery Networks)**

This file is once created in object store it does exist somewhere , suppose in a server of AWS(suppose US) . Suppose lot of people are asking for the same file from all over the world and suppose it is 16 MB file . Since it is long file and have to pass through several wires , it can be slow

So wouldn't it be better if lot of people from india are asking for the file then it is first send to a server in India then it is distributed . **CDNs** let us do this.

Don't distribute directly through the S3 url instead use my CDNs URL and tell the source.



It is cached for a certain duration of time. If these content exist only in one part of world then it would have been very difficult . CDN have POPs(point of presence) CDN example are Cloudfront (we access through Cloudfront url and ask to source of truth (Amazon) s3.aws.mp4)

Object Stores store in single place, CDN are multiple server all accross the world, POP check whether it has the content which user asked if not then go fetch from the source of truth

Rate for AWS :--
Are divided into two part storage and distribution
Storage could be like 2GB : 0.02 dollars and for distribution it could be like for occupying 20GB bandwidth it can be like 0.01 dollars (20GB bandwidth means that 2GB file is accessed by 10 people).

Why we don't need to deploy backend from CDN??
When suppose 1000 people are accessing a Mp4 file they are accessing from the same mp4 file.
But when we hit server for /me endpoint for our details , each users get different information according to their information , hence we can't really cache it.
Because each user want something else.
What do make sense is **Edge Networks**
Basically our backend is deployed in multiple servers across the world, but it data is not cached .
Why don't we have multiple source of truth ? in case of S3 . Why do we even need CDNs can't we store it in different part of the world , the reason is that storing these S3 data is **expensive.**

How to deploy Frontends to AWS

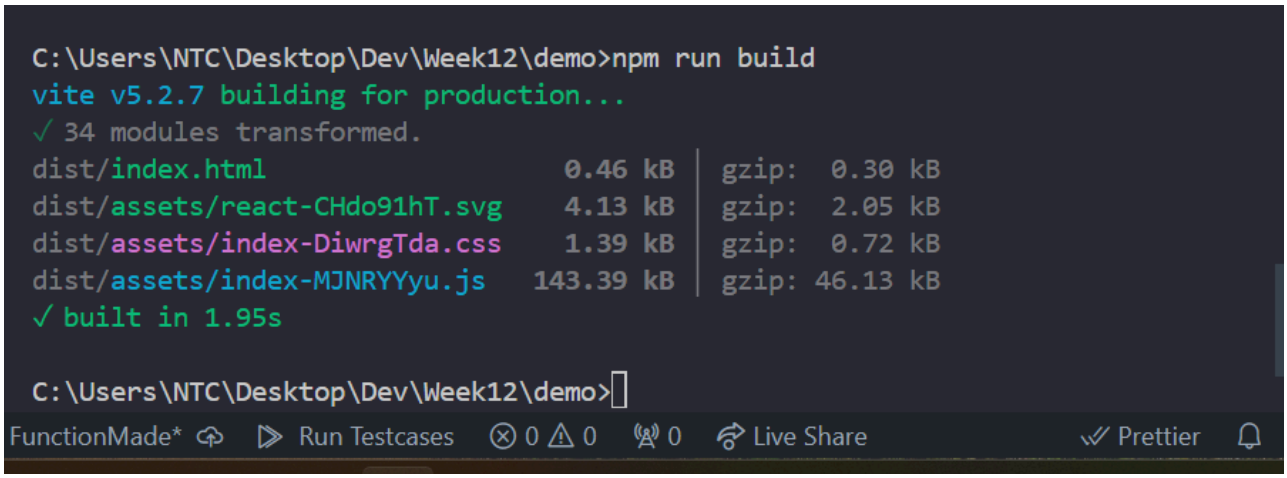
1. Object Stores (S3)
2. CDNs (Cloud front)

Build your React frontend

This approach will not work for frameworks that use Server Side Rendering (Like) next.js This will work for Basic React Apps, HTML/CSS/JS apps

Create a new React project.

We convert our react project into html/css/Js by doing **npm run build**



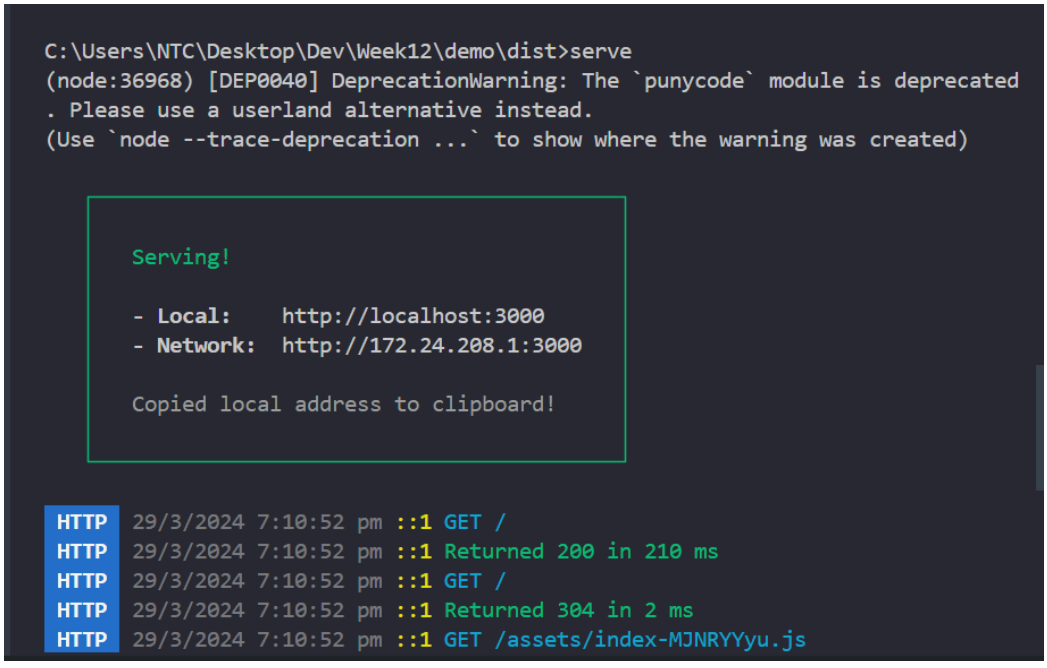
Double Clicking on the index.html we wont be able to see anything

We need slightly cleaner way to distribute

npm install -g serve

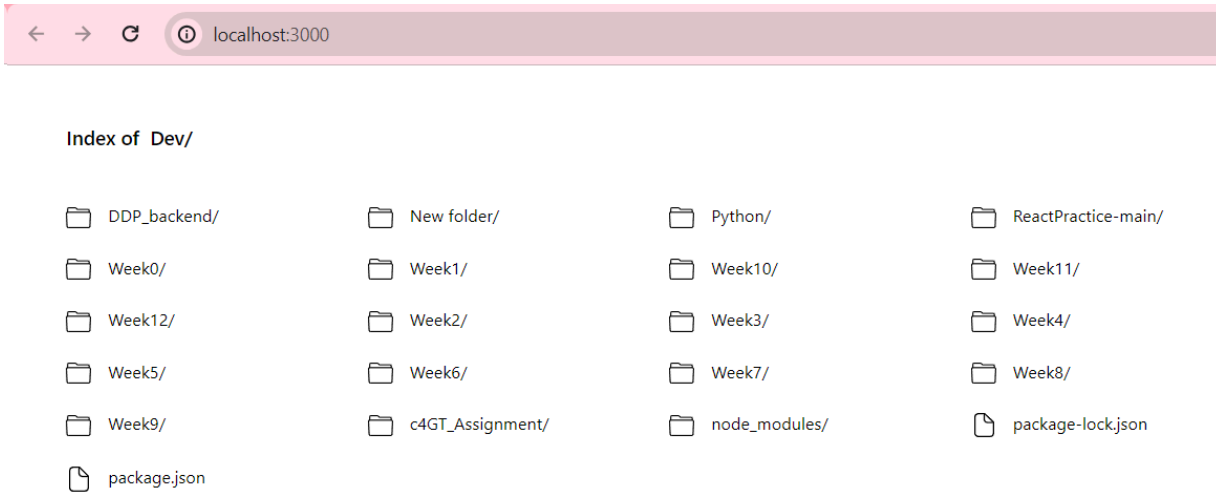
Then we will do **cd dist**

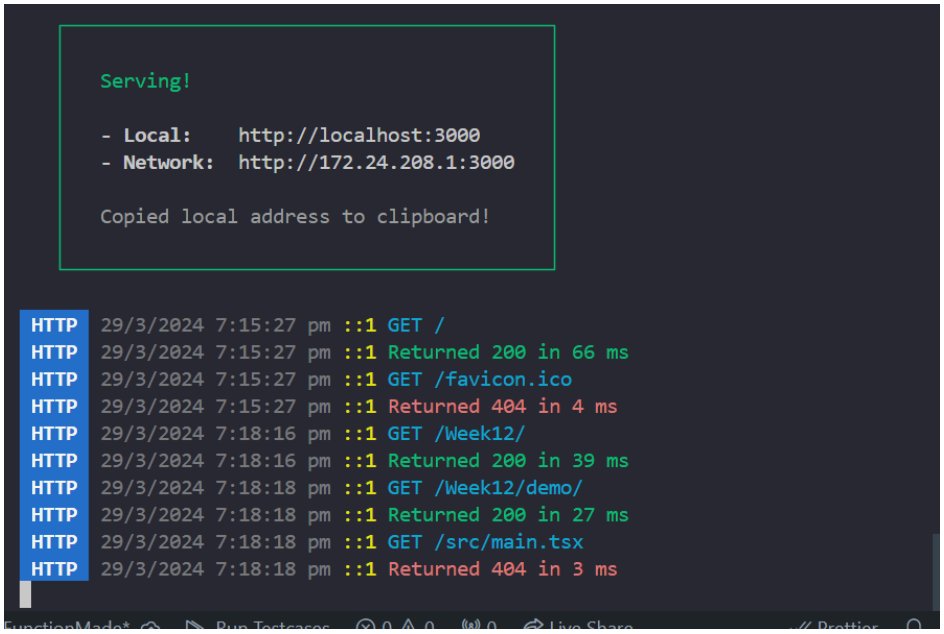
serve



It serves file on a port

If we cd a folder containing lot of files and folder it will basically serve the folder in on a port and suppose someone connect to same Network has that port can access it.





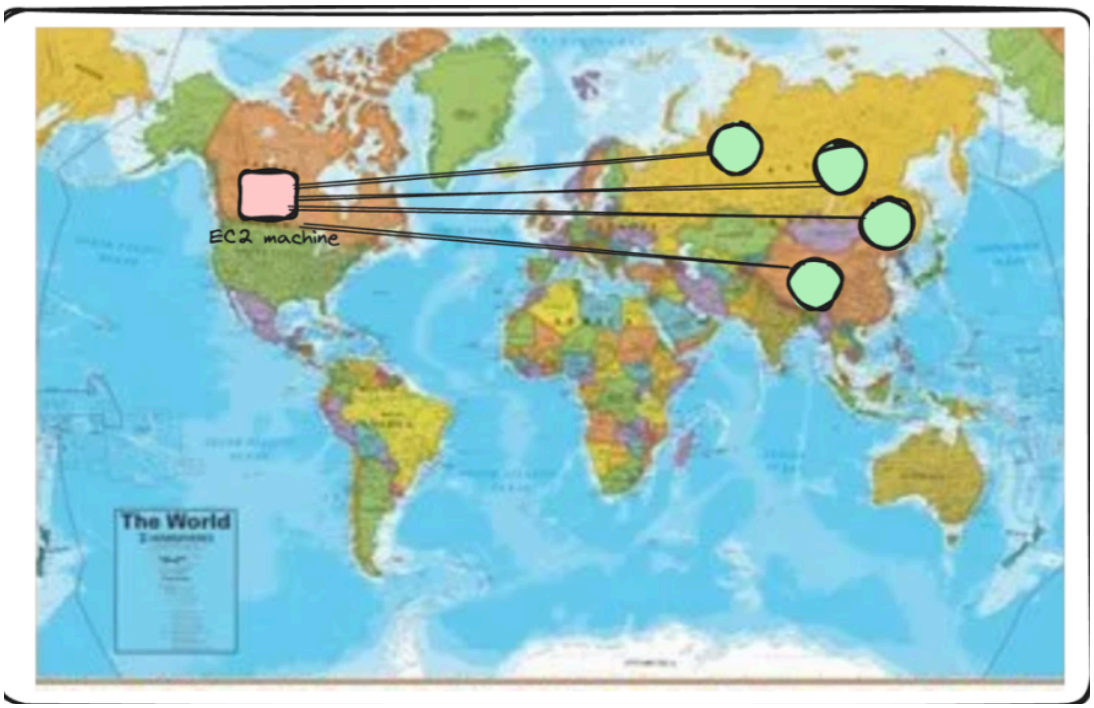
What are CDNs??

A CDN stands for Content Delivery Network.

As the name suggests, it’s an optimal way for you to deliver content (mp4 files, jpgs and even HTML/CSS/JS files) to your users.

It is better than serving it from a VM/EC2 instances because of a few reasons -

1. EC2 machine approach



2. CDN approach

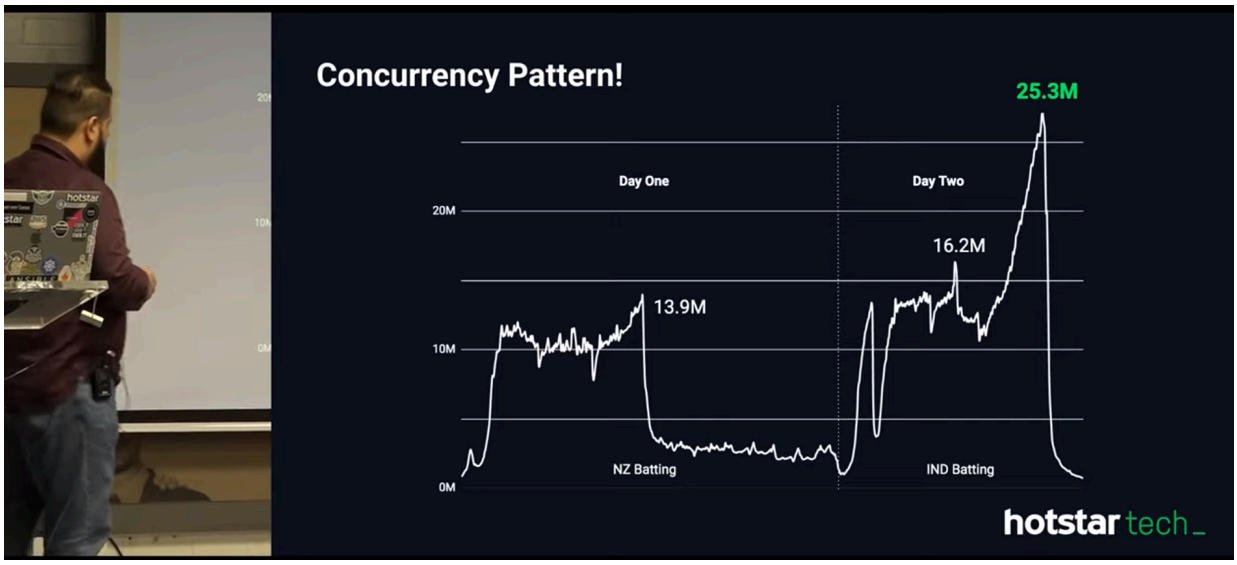
Both these approach are explained before.

1. For frontends, mp4 files, images, Object stores + CDNs are a better approach.
2. You can’t use the same for backends, since every request returns a different response. Caching doesn’t make any sense there.

You can use edge networks for backends (deploy your backend on various servers on the internet) but data can’t be cached in there.

A example will be How Hotstar **scales their infrastructure during cricket matches.**

(they used CDNs heavily)



Spiky is dangerous for backend.

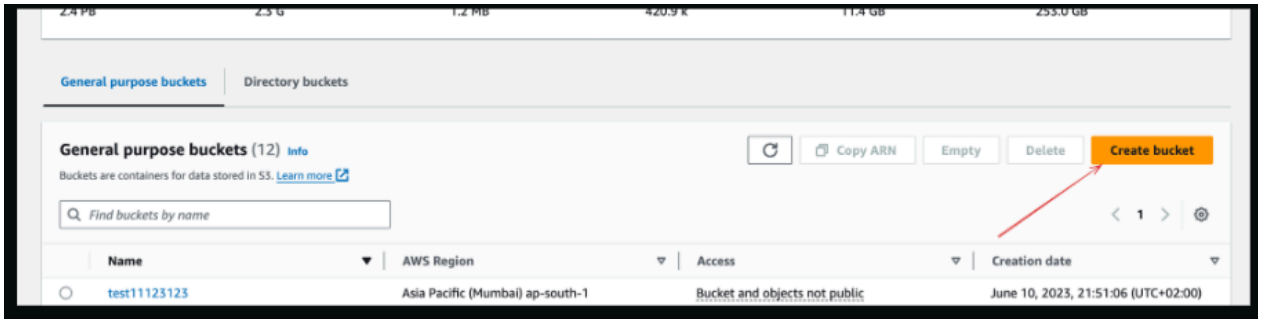
It was around 1milion/min

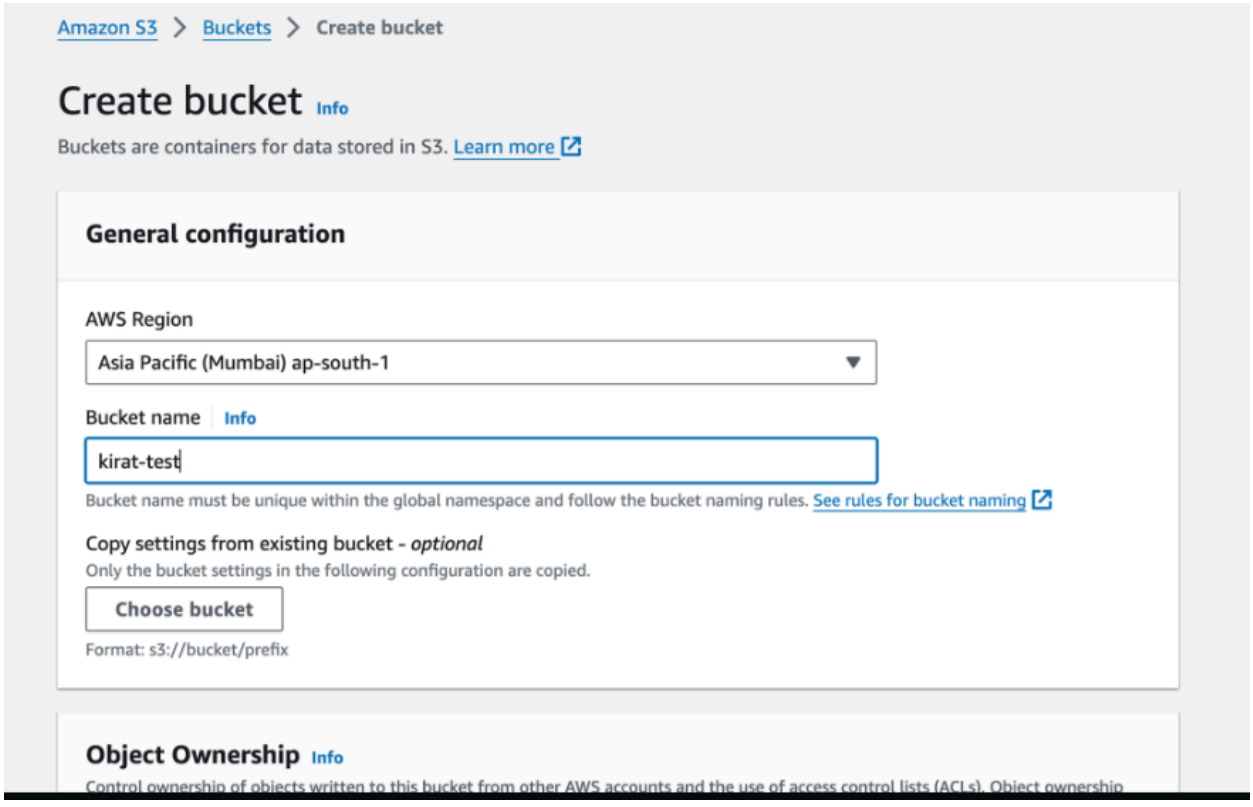
Autoscaling is slow. At 25.3 million Dhoni got out This drops kill most websites. As suddenly there are lot of api calls for **Home page ,** this is something which need to be prepared in advance

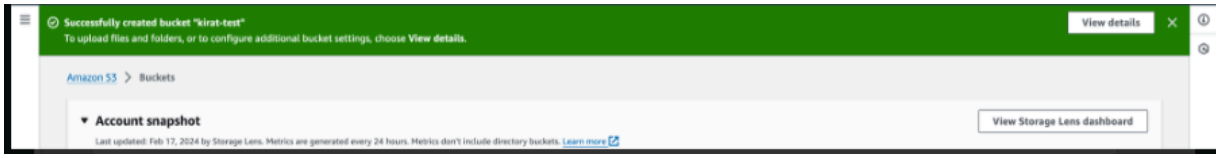
Creating an object store in AWS

In AWS , S3 is their object store offering.

You can create a bucket in there. A bucket represents a logical place where you store all the files of a certain project.

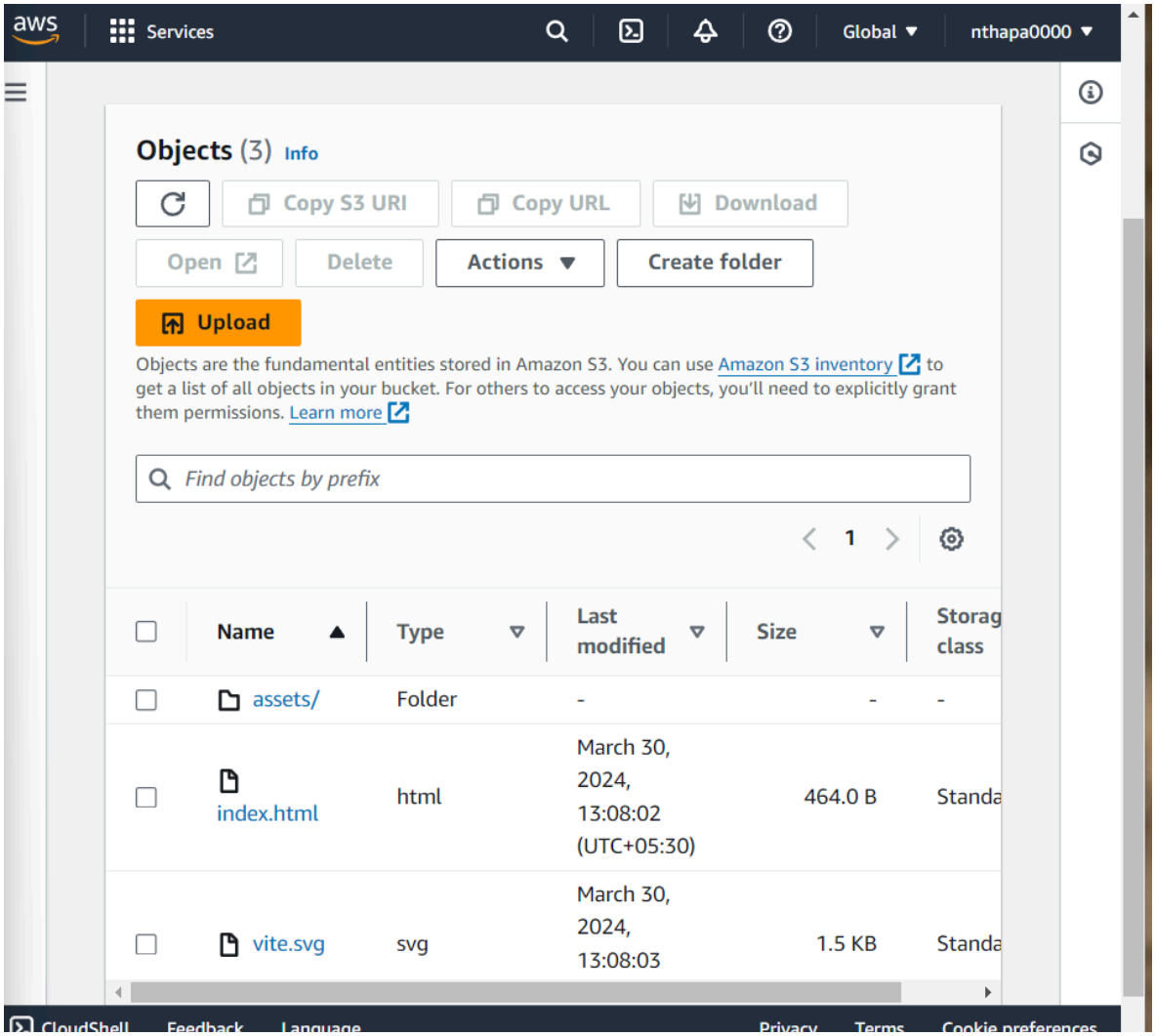




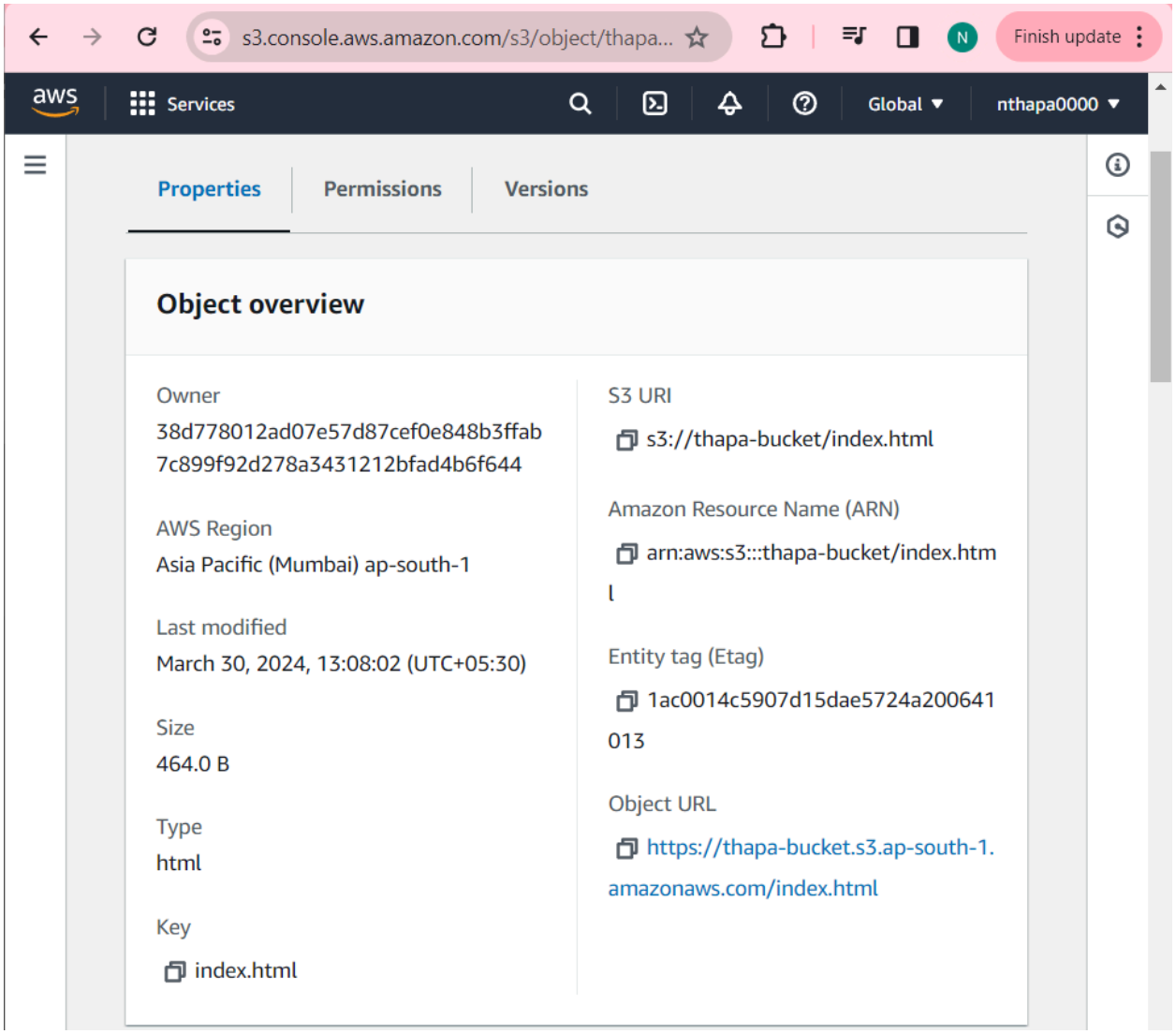


Upload the file/folder bundle

Now upload the dist (production ready) folder inside the S3

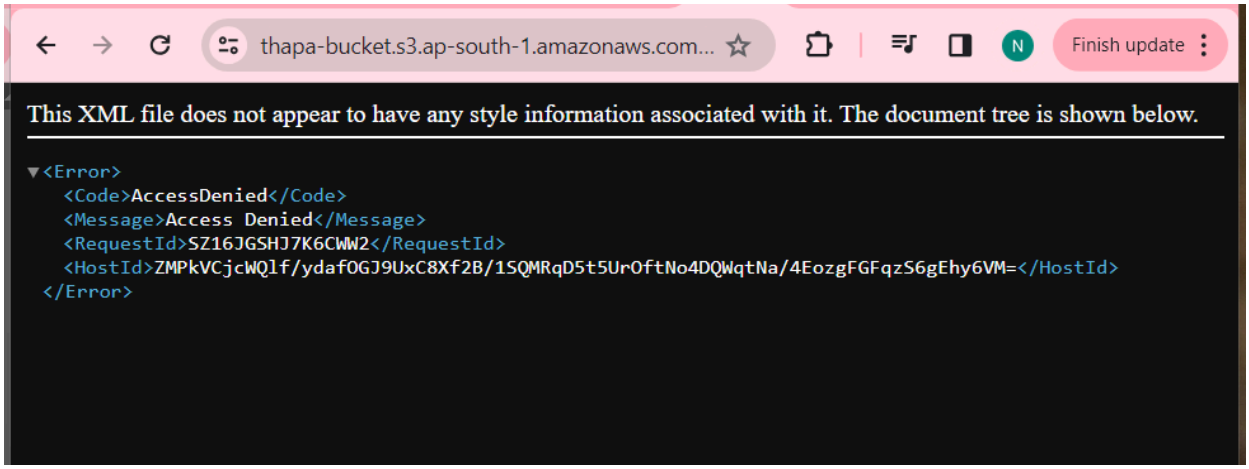


Although if we click on them each have a public url



But we have blocked all the public access (By default)

Try accessing the website/url



We don’t want to distribute data through S3 cause it is expensive and slow

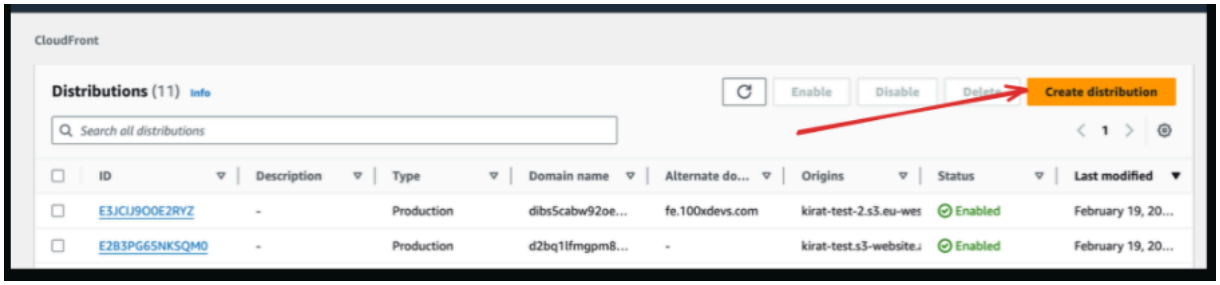
We will go through Cloudfront. **Index.html should be present at root level**

Connecting Cloudfront

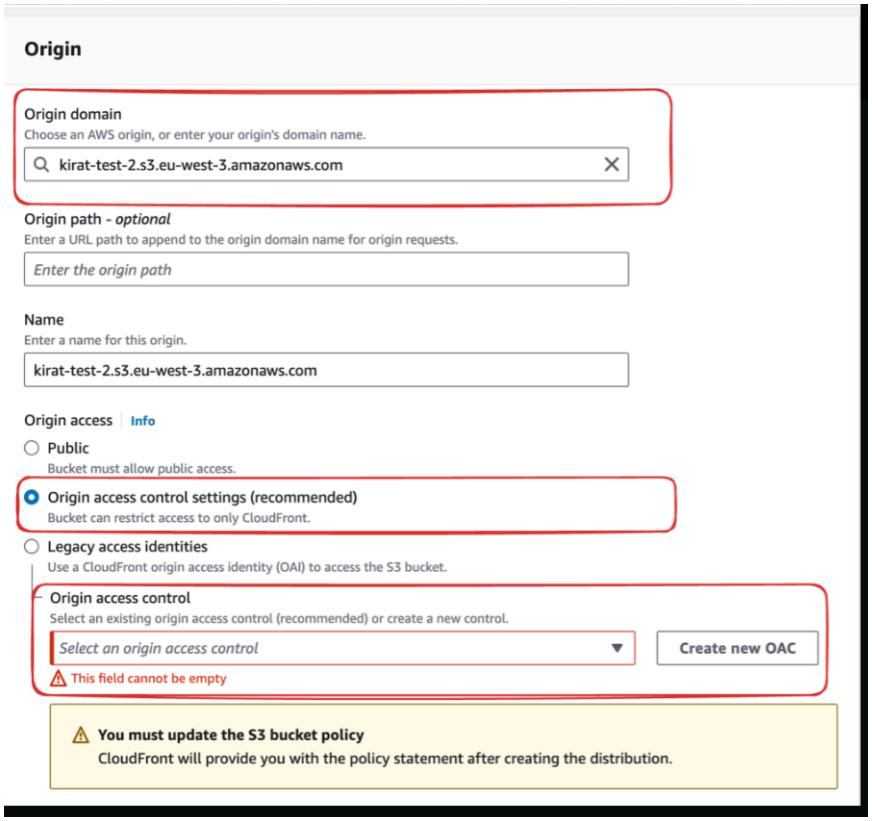
Step 1:-- **Create cloudfront distribution**

Go to cloudfront and create a new distribution. A distribution here means you’re creating a place where content can be distributed.

We can even put google.com



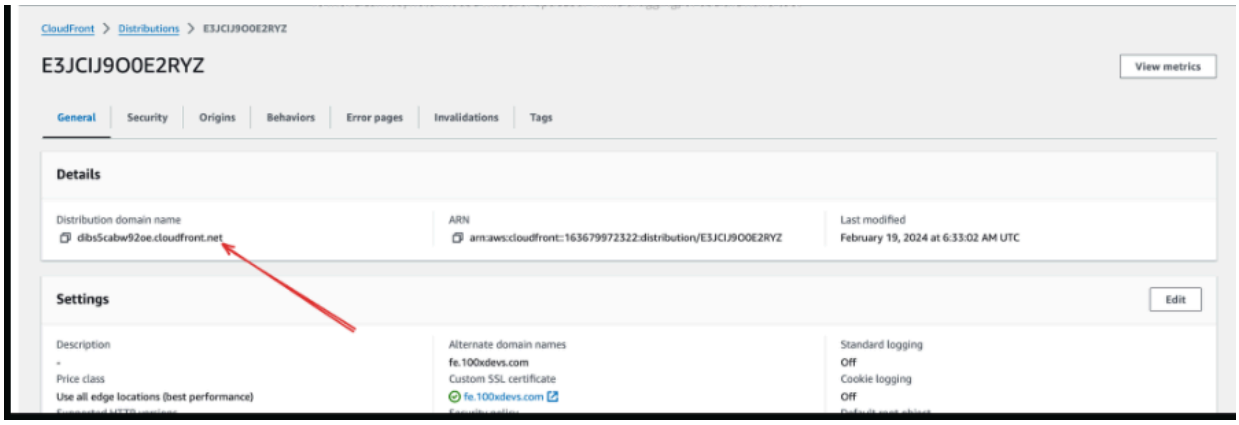
Step 2– **Select your S3 bucket as the source**

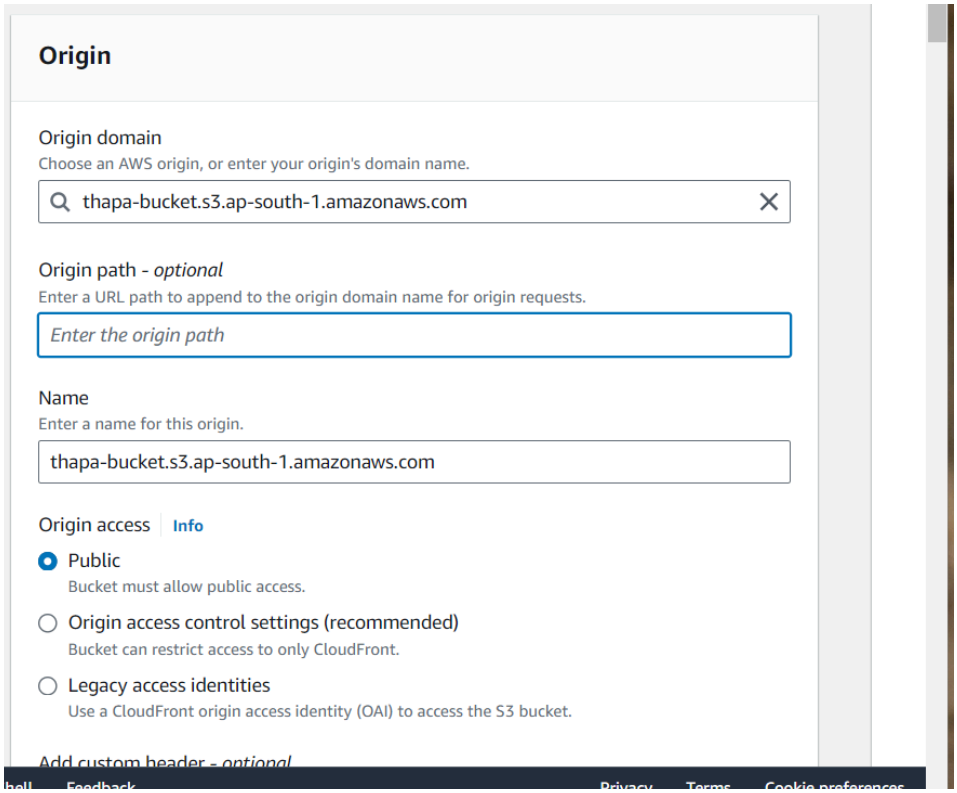


Origin Access Control (OAC) is a feature in Cloudfront, which allows you to restrict direct access to the content stored in your origin, such as an Amazon S3 bucket or a web server, ensuring that users can only access the content through the CDN distribution and not by directly accessing the origin URL.

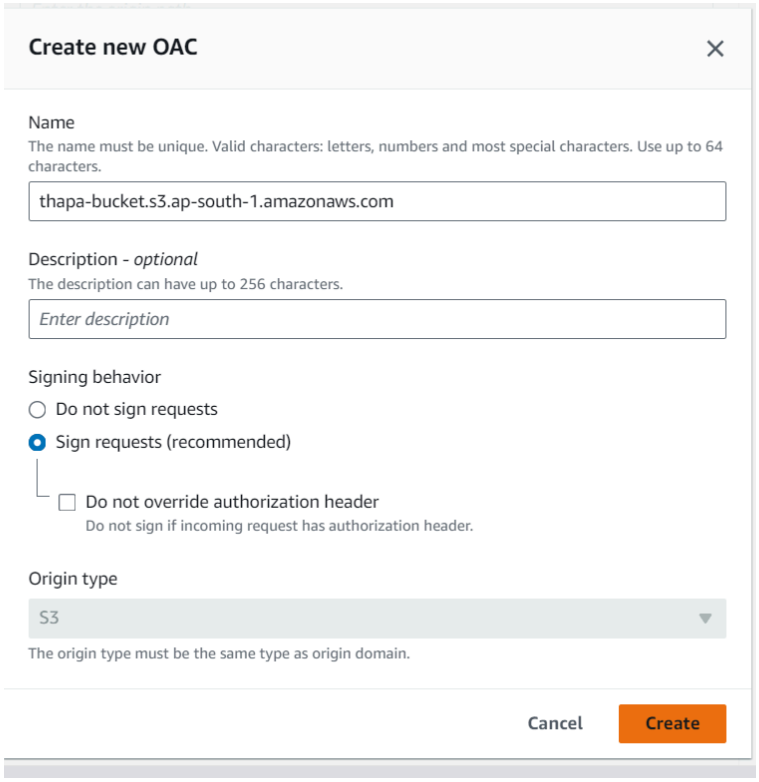
Basically how the cloudfront can access our origin

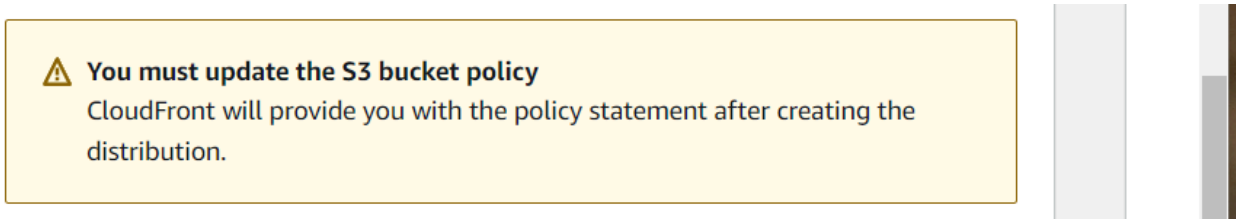
By the end of this you should have a working cloudfront URL



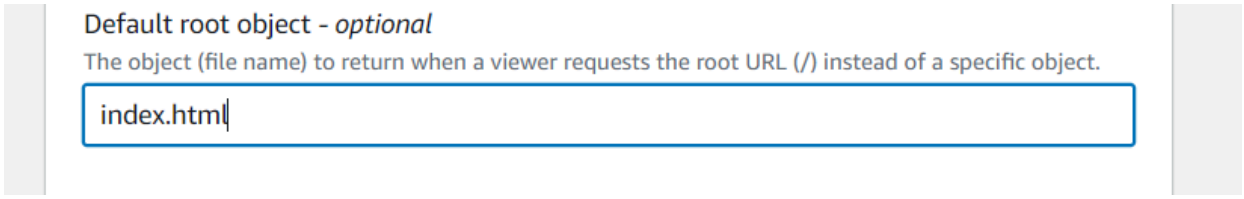


If we have by mistakenly uploaded the dist folder then we can give origin path



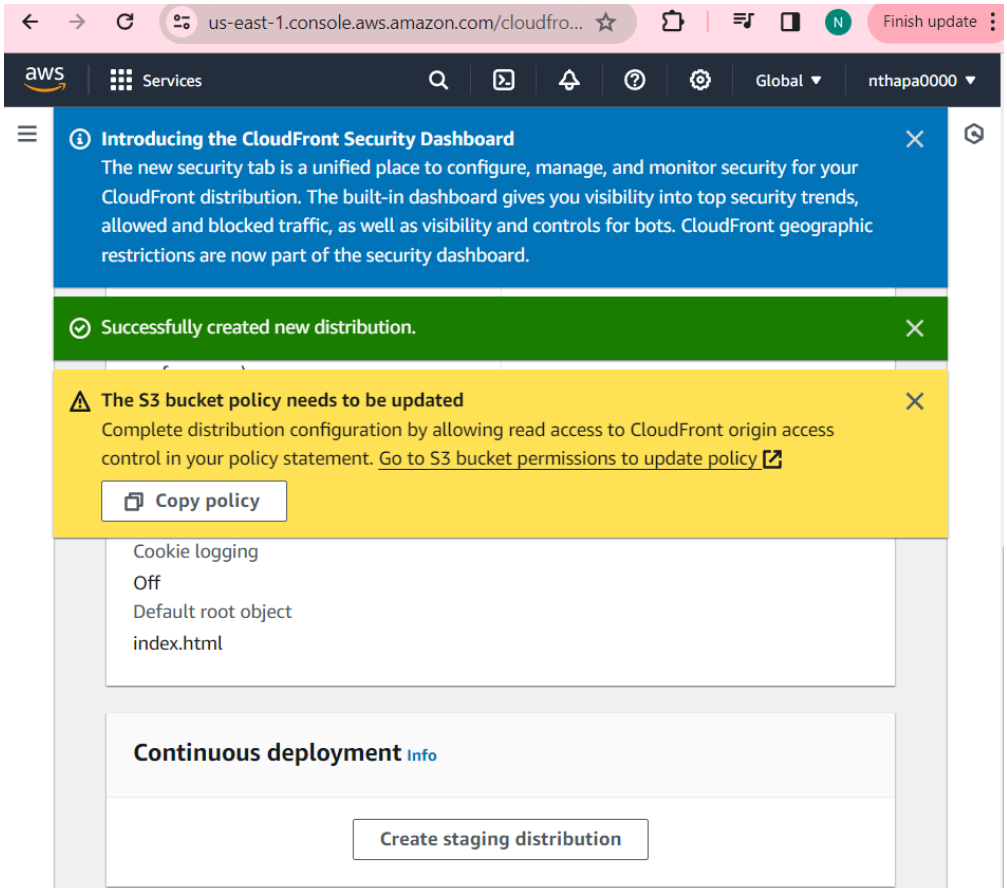


Basically at end we wil get a file and we have to put it in S3

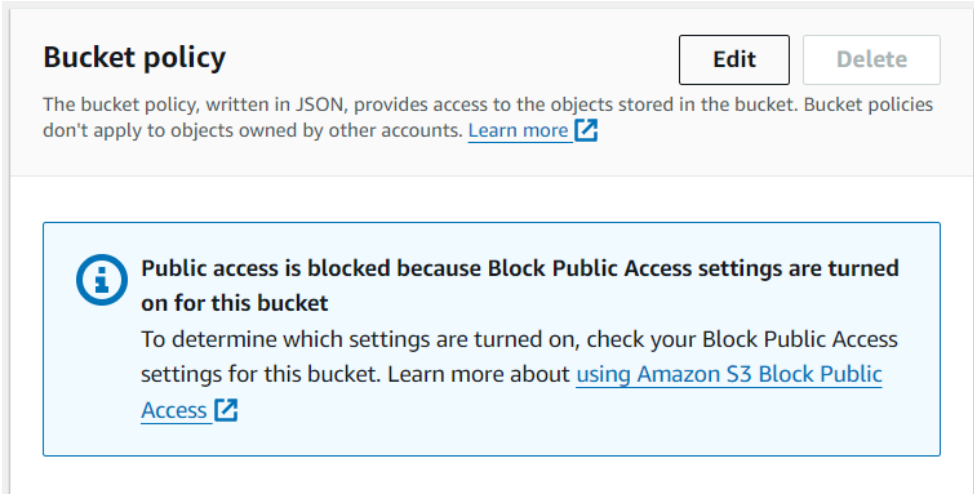


By default the file which should be picked is index.html when we go to the root.

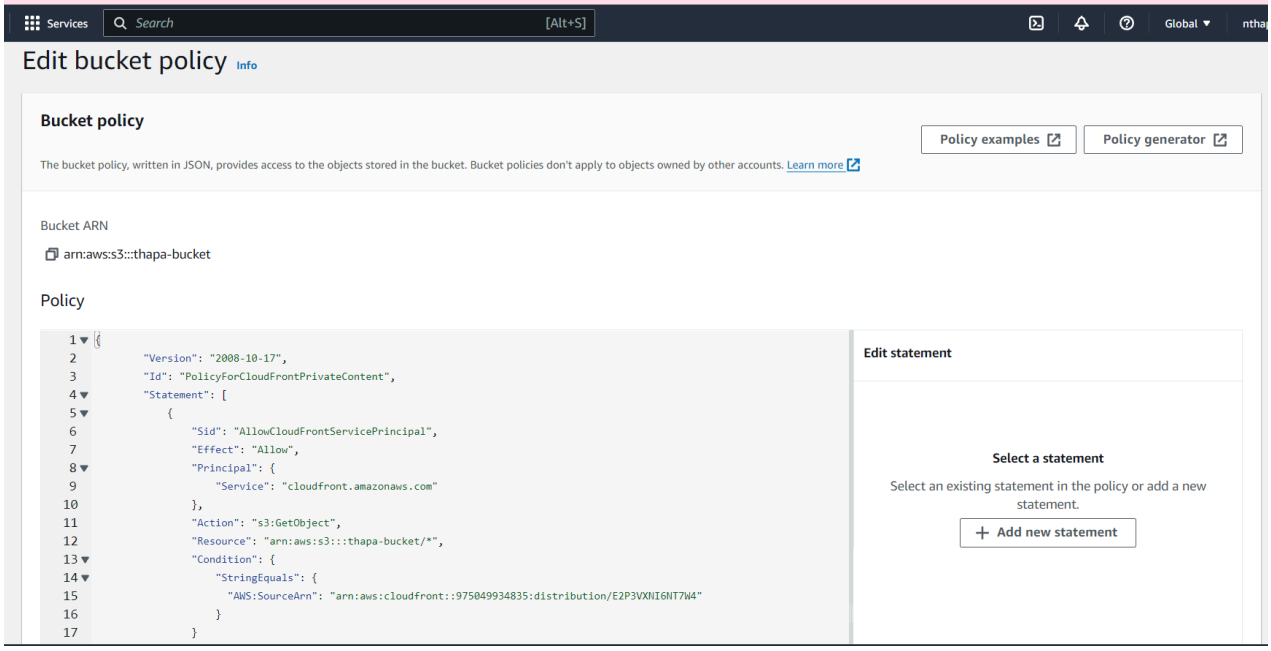
Now we need to update S3 policy



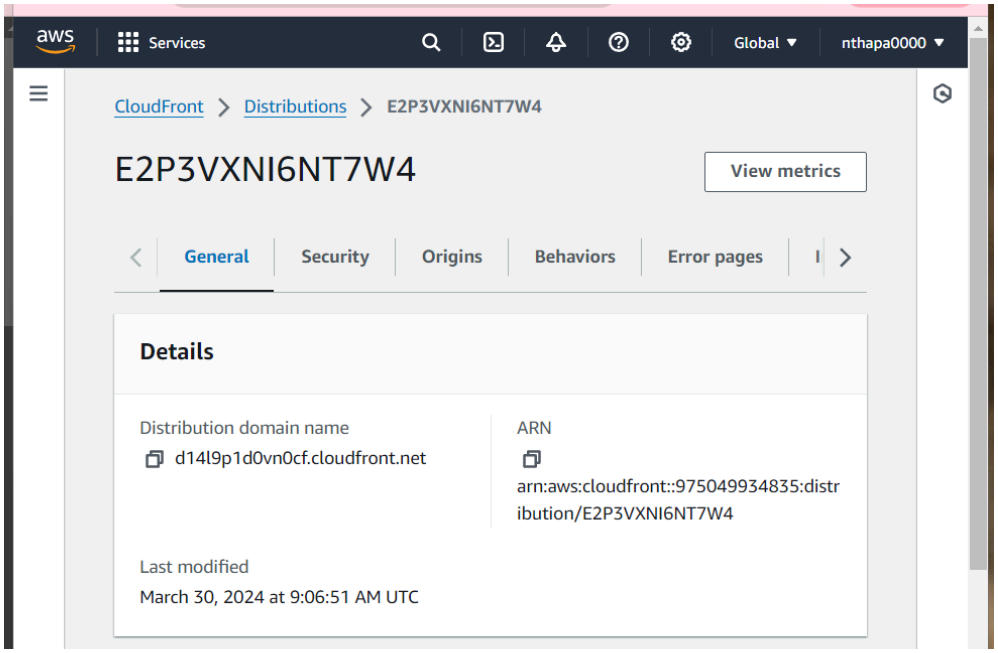
We will copy the policy



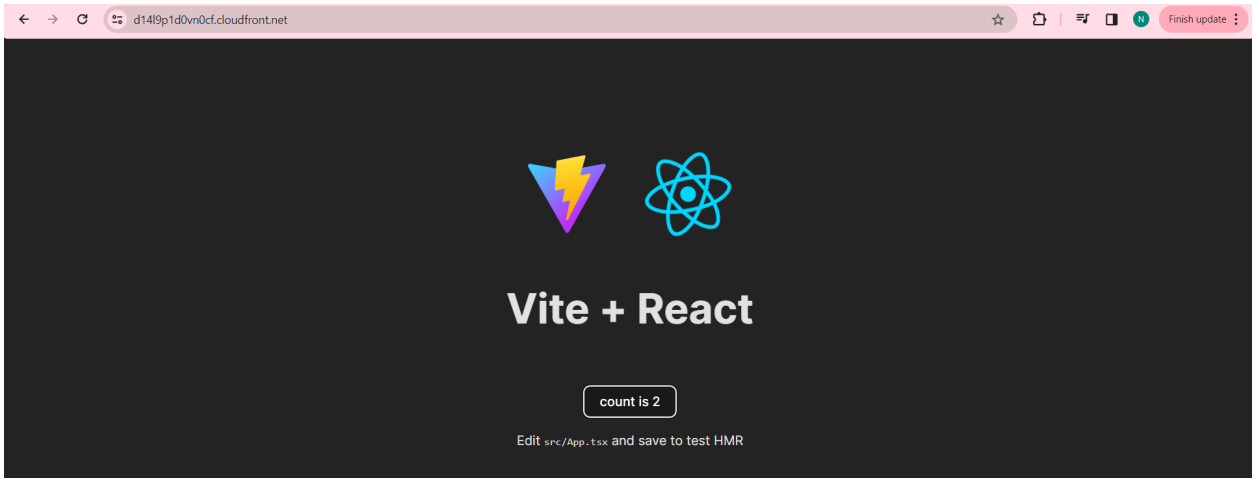
Click on Edit and paste the policy



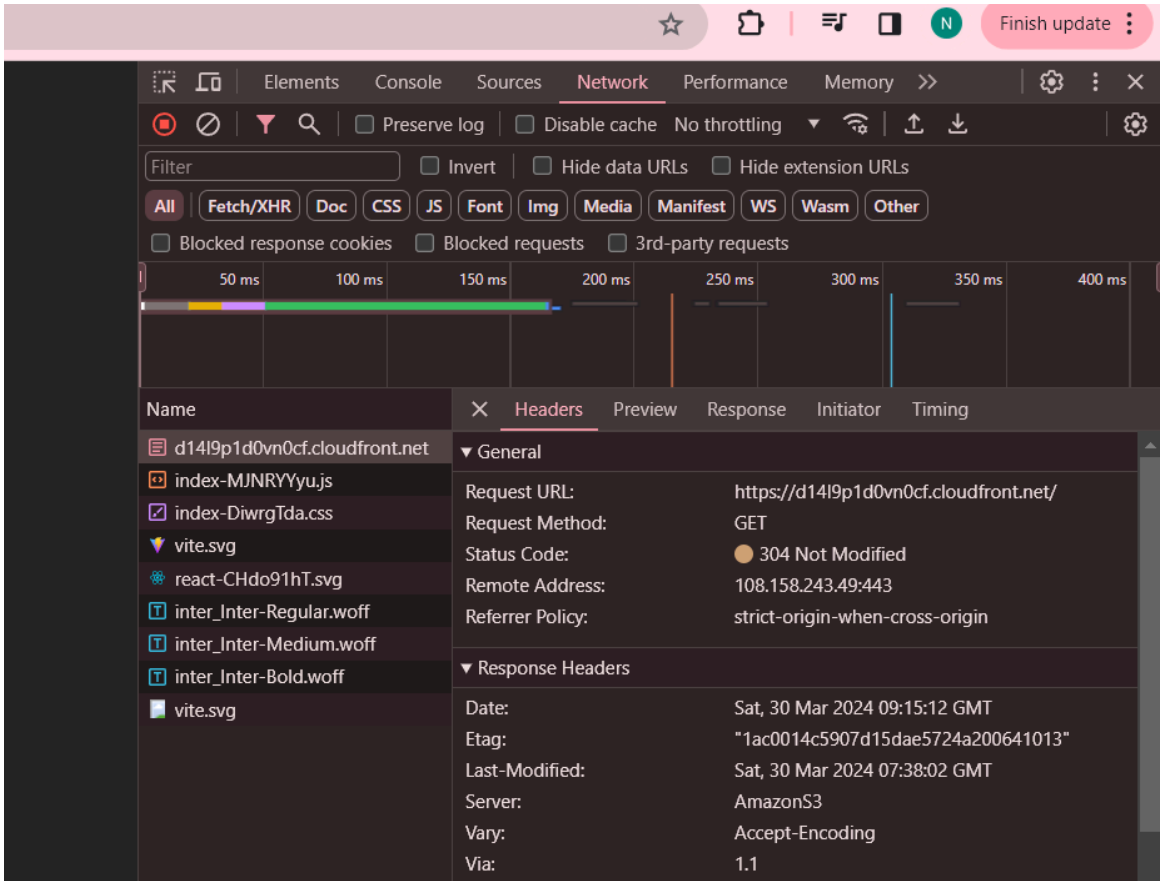
Policy generate says that anything inside S3 bucket should be allowed by this specific Cloudfront distribution.



Now only problem is that we can say is that we want to give out a pretty url to user. Distribution domain name will get us to our app



Network tab we can see the Request URL

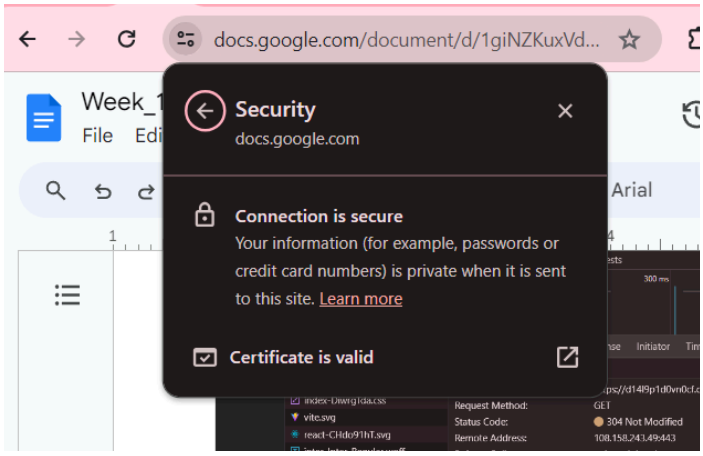


Connect you own domain to website

Any frontend should be deployed in https

Custom ssl certificate.

Whenever we go to website we can see that certificate is valid or not.



We can now deploy our own domain

Add an error page