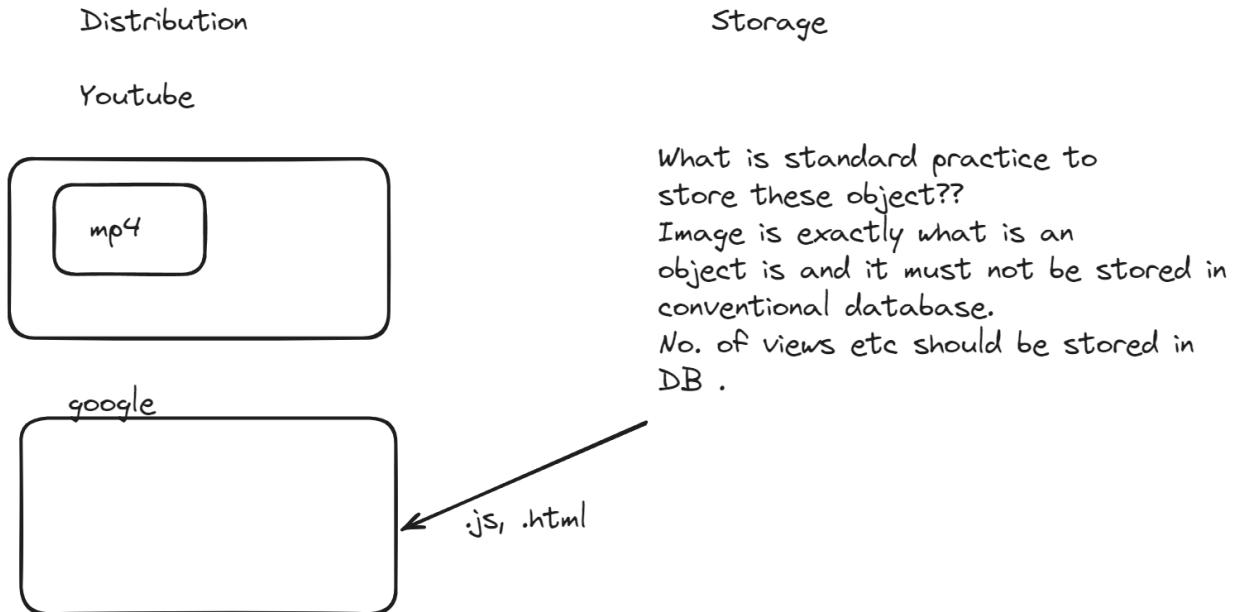
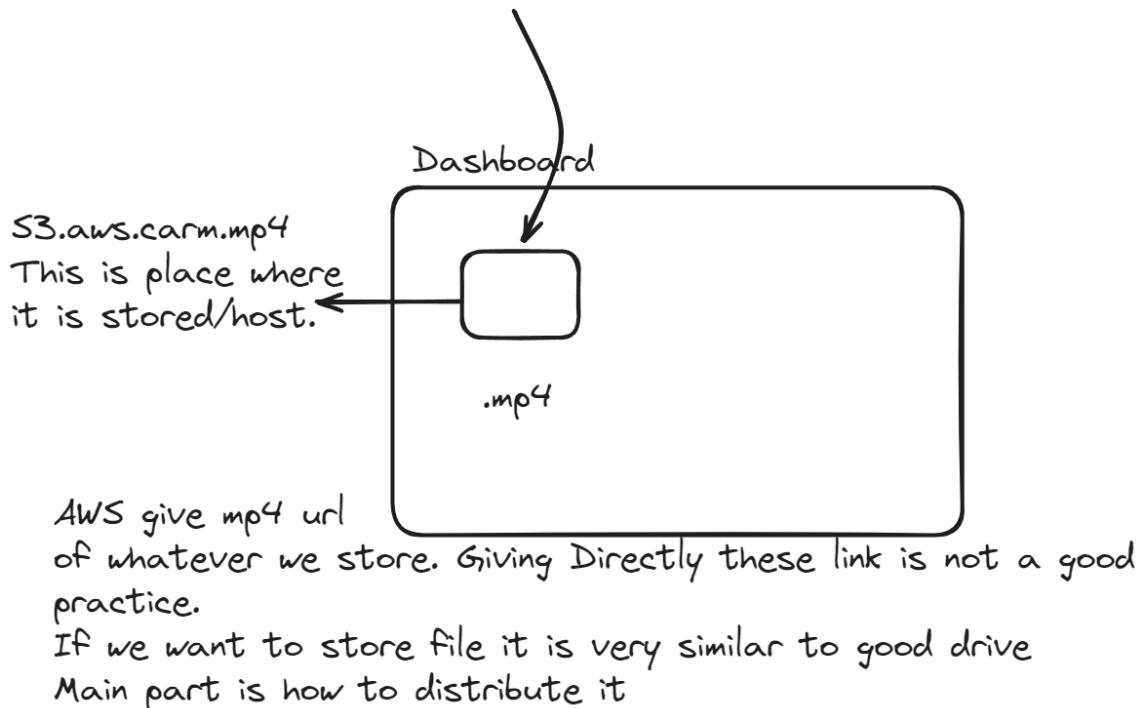


Deploying Frontend on AWS

(<https://projects.100xdevs.com/tracks/w5E6PT2t0lyOFM3bZxcM/aws-fe-1>)



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



We need to know about **CDNs** .

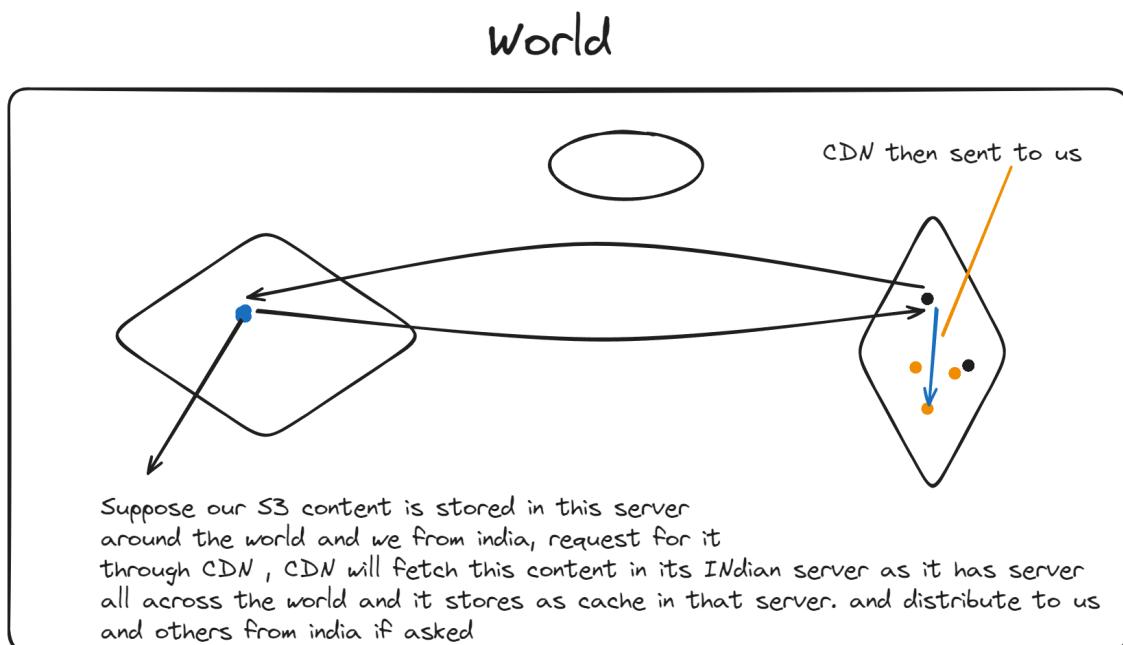
Storage — object stores

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**

```
C:\Users\NTC\Desktop\Dev\Week12\demo>npm run build
vite v5.2.7 building for production...
✓ 34 modules transformed.
dist/index.html          0.46 kB  gzip:  0.30 kB
dist/assets/react-CHdo91hT.svg 4.13 kB  gzip:  2.05 kB
dist/assets/index-DiwrgTda.css 1.39 kB  gzip:  0.72 kB
dist/assets/index-MJNRYYyu.js 143.39 kB  gzip: 46.13 kB
✓ built in 1.95s

C:\Users\NTC\Desktop\Dev\Week12\demo>[]
```

FunctionMade* 0 0

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

```
C:\Users\NTC\Desktop\Dev\Week12\demo\dist>serve
(node:36968) [DEP0040] DeprecationWarning: The `punycode` module is deprecated
. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)

Serving!
- Local:   http://localhost:3000
- Network: http://172.24.208.1:3000

Copied local address to clipboard!
```

```
HTTP 29/3/2024 7:10:52 pm ::1 GET /
HTTP 29/3/2024 7:10:52 pm ::1 Returned 200 in 210 ms
HTTP 29/3/2024 7:10:52 pm ::1 GET /
HTTP 29/3/2024 7:10:52 pm ::1 Returned 304 in 2 ms
HTTP 29/3/2024 7:10:52 pm ::1 GET /assets/index-MJNRYYyu.js
```

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.



```
Serving!
- Local: http://localhost:3000
- Network: http://172.24.208.1:3000
Copied local address to clipboard!

HTTP 29/3/2024 7:15:27 pm ::1 GET /
HTTP 29/3/2024 7:15:27 pm ::1 Returned 200 in 66 ms
HTTP 29/3/2024 7:15:27 pm ::1 GET /favicon.ico
HTTP 29/3/2024 7:15:27 pm ::1 Returned 404 in 4 ms
HTTP 29/3/2024 7:18:16 pm ::1 GET /Week12/
HTTP 29/3/2024 7:18:16 pm ::1 Returned 200 in 39 ms
HTTP 29/3/2024 7:18:18 pm ::1 GET /Week12/demo/
HTTP 29/3/2024 7:18:18 pm ::1 Returned 200 in 27 ms
HTTP 29/3/2024 7:18:18 pm ::1 GET /src/main.tsx
HTTP 29/3/2024 7:18:18 pm ::1 Returned 404 in 3 ms
```

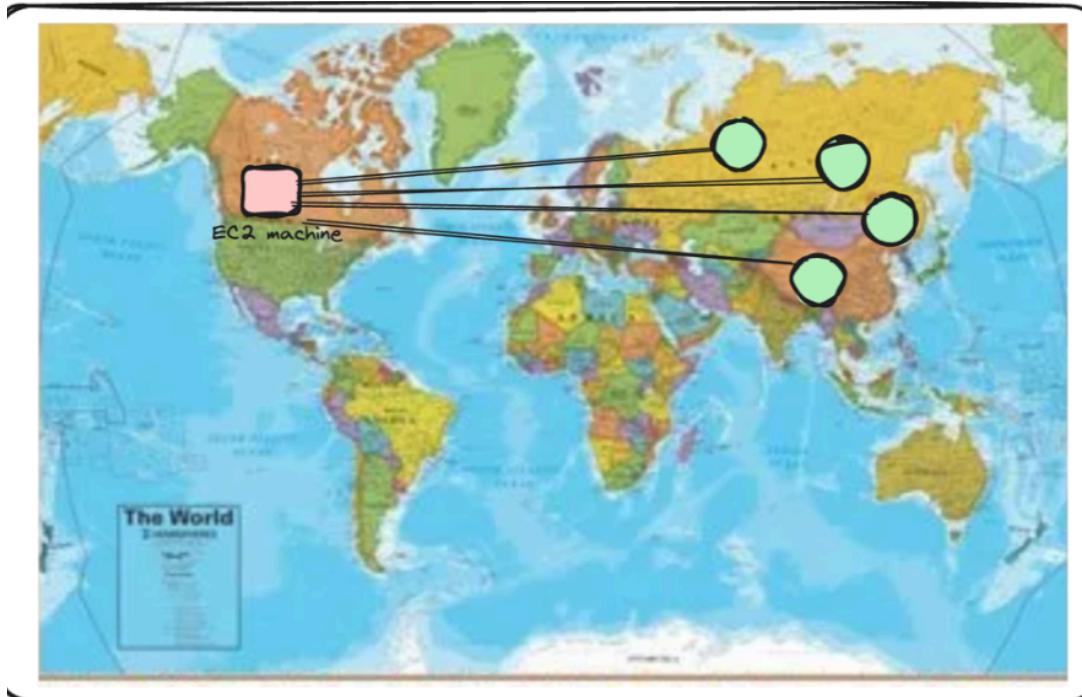
What are CDNs??

A CDN stands for Content Delivery Network.

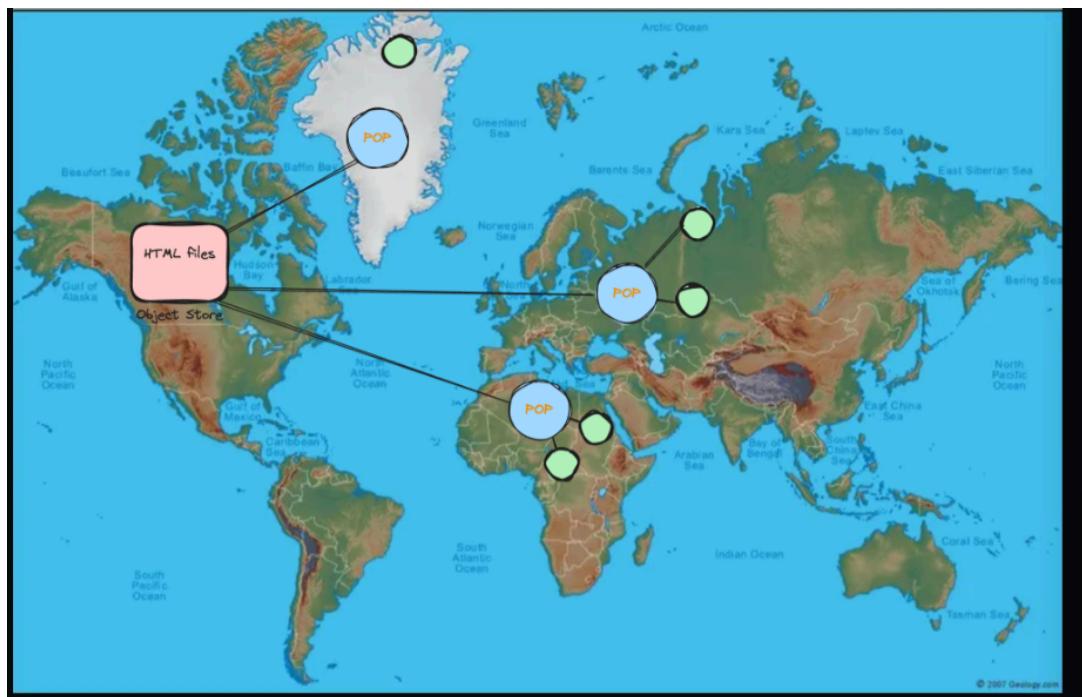
As the name suggests, it's an optimal way for you to deliver content (mp4 files, jpgs and even HTML/CSS/JS files) to your users.

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

1. EC2 machine approach



2. CDN approach



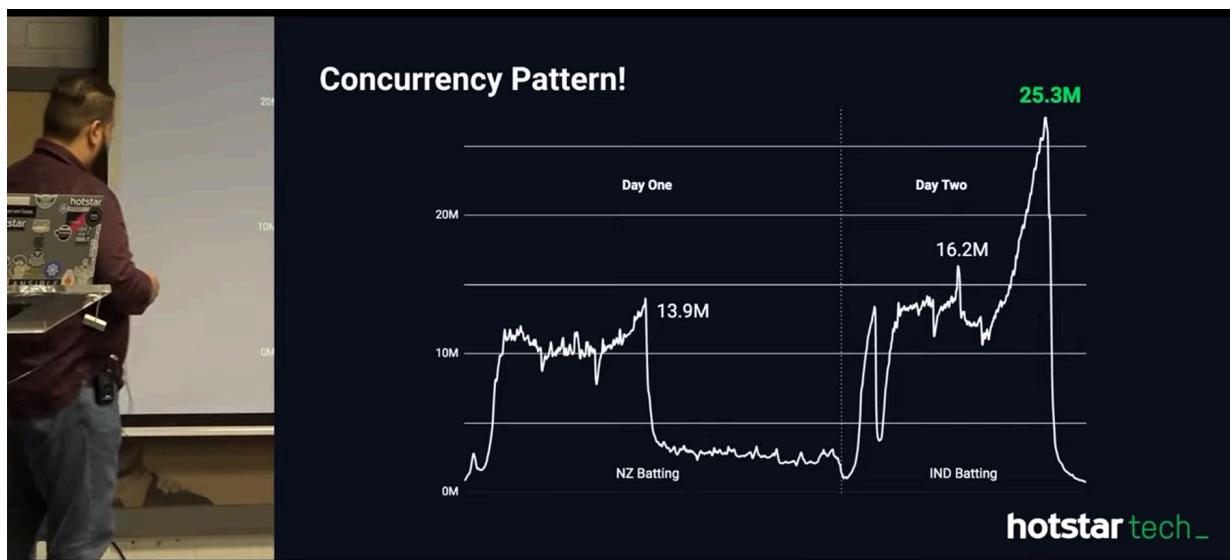
Both these approach are explained before.

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.

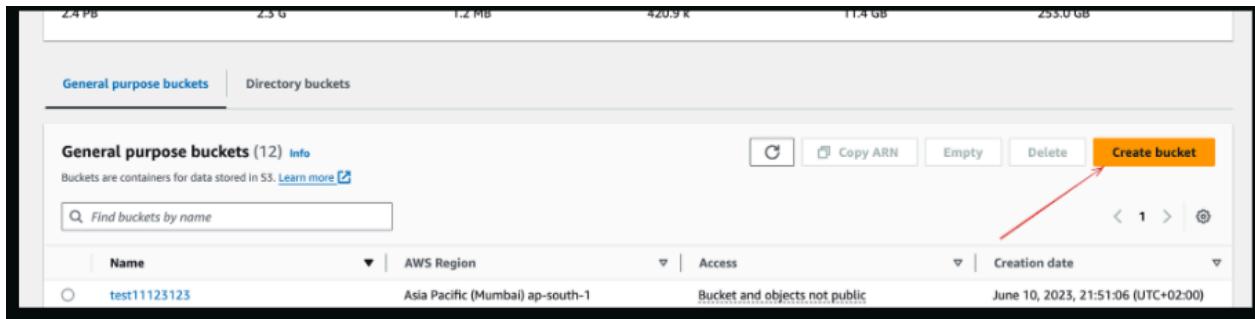
It was around 1million/min

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

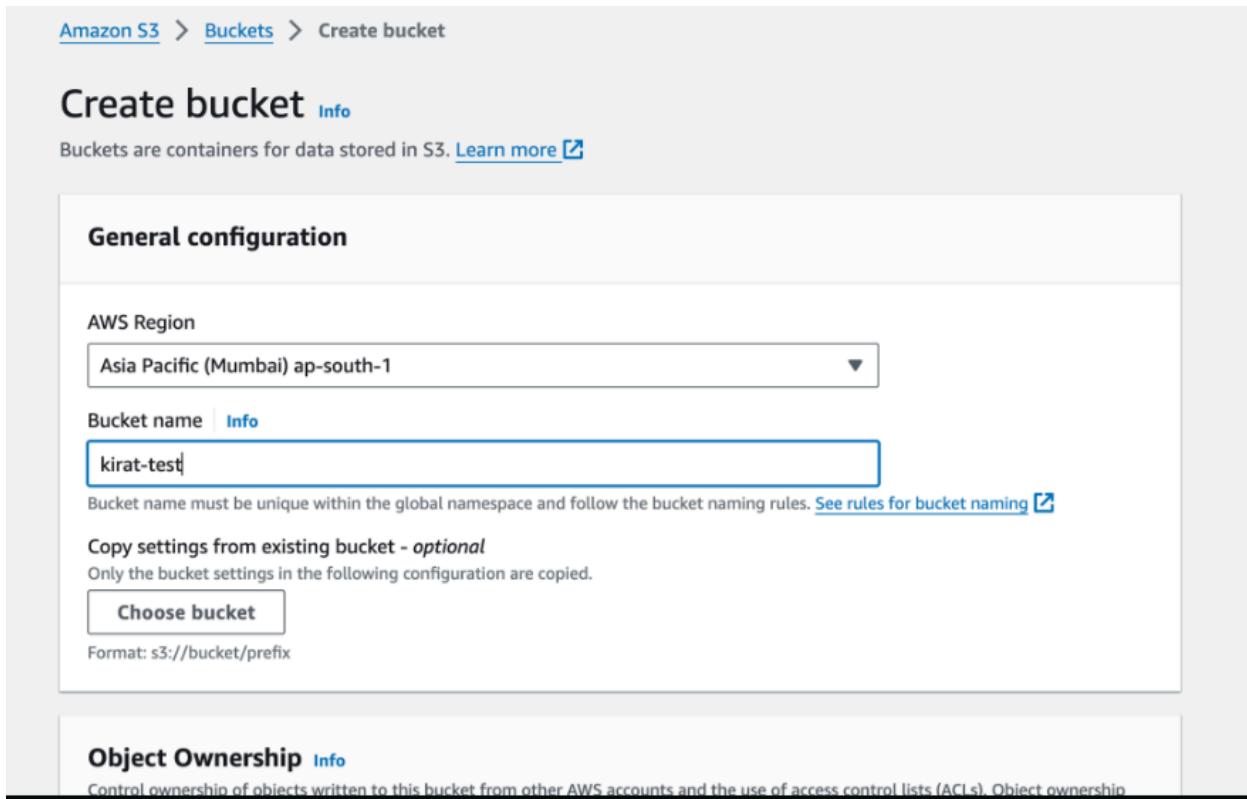
Creating an object store in AWS

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.



The screenshot shows the AWS S3 console interface. At the top, there are several status indicators: 2.4 PB, 2.5 G, 1.2 MB, 420.9 K, 11.4 GB, and 255.0 GB. Below these, there are two tabs: 'General purpose buckets' (selected) and 'Directory buckets'. A search bar labeled 'Find buckets by name' is present. To the right of the search bar are buttons for 'Copy ARN', 'Empty', 'Delete', and a prominent orange 'Create bucket' button. Below these buttons is a pagination control showing page 1 of 1. The main table lists one bucket: 'test11123123' located in 'Asia Pacific (Mumbai) ap-south-1'. The table columns are 'Name', 'AWS Region', 'Access', and 'Creation date'. The 'Access' column shows 'Bucket and objects not public' and the 'Creation date' shows 'June 10, 2023, 21:51:06 (UTC+02:00)'.



The screenshot shows the 'Create bucket' wizard. The first step, 'General configuration', is displayed. It includes fields for 'AWS Region' (set to 'Asia Pacific (Mumbai) ap-south-1'), 'Bucket name' (set to 'kirat-test'), and a note about naming rules. Below these are sections for 'Copy settings from existing bucket - optional' (with a 'Choose bucket' button) and 'Format: s3://bucket/prefix'. The second step, 'Object Ownership', is partially visible at the bottom.

The screenshot shows a green success message at the top: "Successfully created bucket 'kirat-test'. To upload files and folders, or to configure additional bucket settings, choose View details." Below this, there's an "Account snapshot" section with a note about last update and a "View Storage Lens dashboard" button. The main navigation bar includes "Amazon S3 > Buckets".

Upload the file/folder bundle

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

The screenshot shows the AWS S3 Objects list view. The top navigation bar includes "aws", "Services", search, notifications, global dropdown, and user info. The main area shows "Objects (3) Info" with buttons for "Copy S3 URI", "Copy URL", "Download", "Open", "Delete", "Actions", and "Create folder". A prominent orange "Upload" button is visible. A descriptive text block explains objects and their storage. A search bar allows finding objects by prefix. The table lists three objects:

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	assets/	Folder	-	-	-
<input type="checkbox"/>	index.html	html	March 30, 2024, 13:08:02 (UTC+05:30)	464.0 B	Standard
<input type="checkbox"/>	vite.svg	svg	March 30, 2024, 13:08:03	1.5 KB	Standard

Although if we click on them each have a public url

The screenshot shows the AWS S3 Object Overview page for an object named 'index.html'. The top navigation bar includes links for 'Properties', 'Permissions', and 'Versions'. The main content area is titled 'Object overview' and displays the following details:

Attribute	Value
Owner	38d778012ad07e57d87cef0e848b3ffab7c899f92d278a3431212bfad4b6f644
AWS Region	Asia Pacific (Mumbai) ap-south-1
Last modified	March 30, 2024, 13:08:02 (UTC+05:30)
Size	464.0 B
Type	html
Key	index.html
S3 URI	s3://thapa-bucket/index.html
Amazon Resource Name (ARN)	arn:aws:s3:::thapa-bucket/index.htm
Entity tag (Etag)	1ac0014c5907d15dae5724a200641013
Object URL	https://thapa-bucket.s3.ap-south-1.amazonaws.com/index.html

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

Try accessing the website/url

The screenshot shows a web browser displaying an XML error response from the URL 'thapa-bucket.s3.ap-south-1.amazonaws.com'. The error message is as follows:

```
<Error>
  <Code>AccessDenied</Code>
  <Message>Access Denied</Message>
  <RequestId>SZ16JGSHJ7K6CWN2</RequestId>
  <HostId>ZMPkVCjclWQ1f/ydaf0GJ9UxC8Xf2B/1SQMRqD5t5UrOftNo4DQWqtNa/4EozgFGFqzS6gEhy6VM=</HostId>
</Error>
```

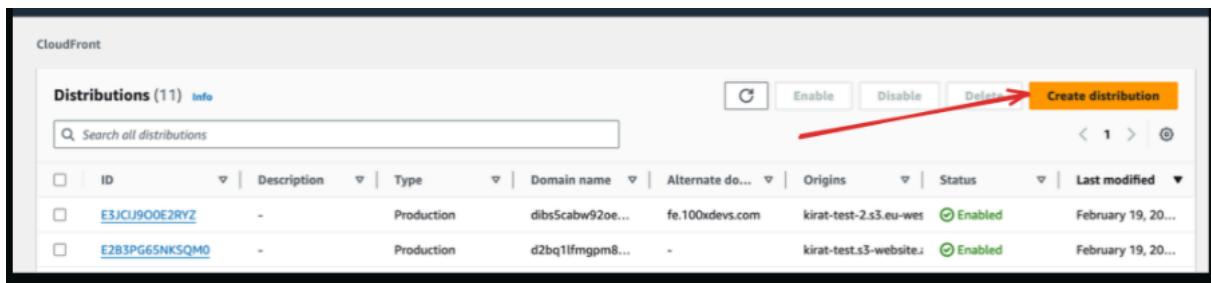
We don't want to distribute data through S3 cause it is expensive and slow
We will go through Cloudfront. **Index.html should be present at root level**

Connecting Cloudfront

Step 1:-- Create cloudfront distribution

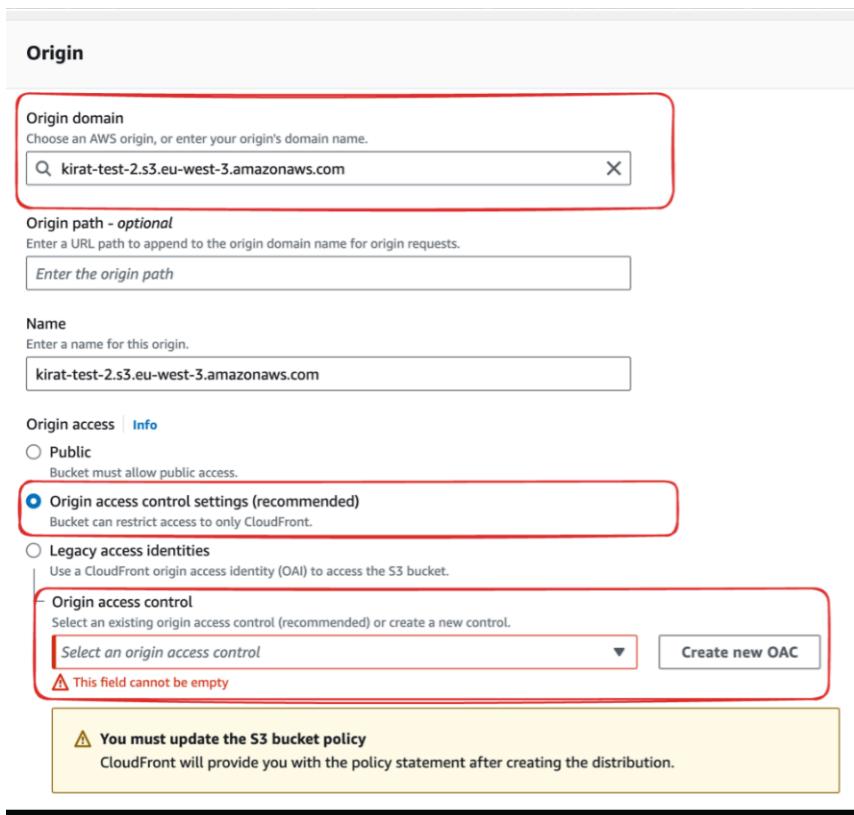
Go to cloudfont and create a new distribution. A distribution here means you're creating a place where content can be distributed.

We can even put google.com



The screenshot shows the AWS CloudFront Distributions page. It displays a table with 11 distributions. The columns include ID, Description, Type, Domain name, Alternate do..., Origins, Status, and Last modified. Two distributions are visible: one with ID E3JCLJ900E2RYZ and another with ID E2B3PG6SNKSQMO. Both are listed as Production type. The Origins column shows the respective S3 buckets. The Status column indicates they are Enabled. The last column shows the date February 19, 2023. A red arrow points to the 'Create distribution' button at the top right of the table.

Step 2– Select your S3 bucket as the source

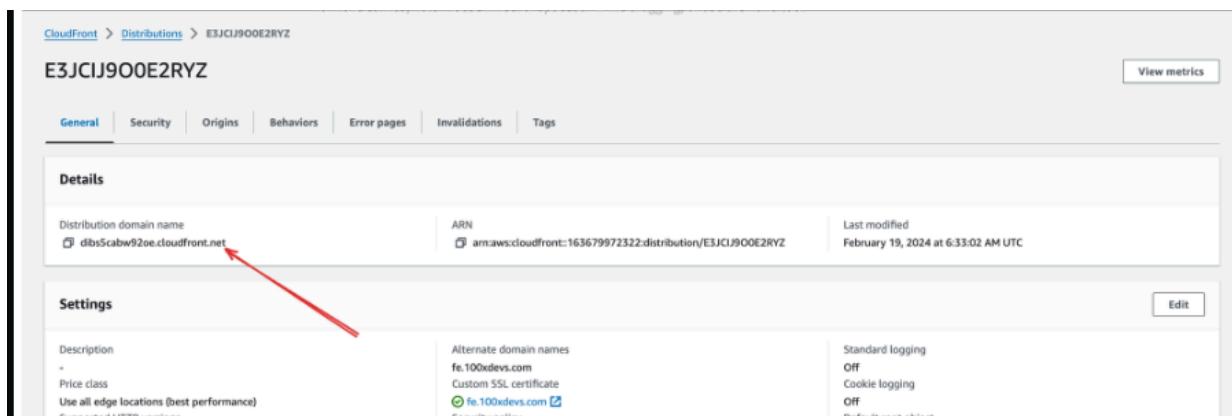


The screenshot shows the 'Origin' configuration page. It includes fields for 'Origin domain' (set to kirat-test-2.s3.eu-west-3.amazonaws.com), 'Origin path - optional' (empty), 'Name' (set to kirat-test-2.s3.eu-west-3.amazonaws.com), and 'Origin access'. Under 'Origin access', the 'Origin access control settings (recommended)' option is selected, which is highlighted with a red box. Below this, there is a dropdown for 'Select an origin access control' and a 'Create new OAC' button. A warning message at the bottom states: '⚠ You must update the S3 bucket policy' and 'CloudFront will provide you with the policy statement after creating the distribution.'

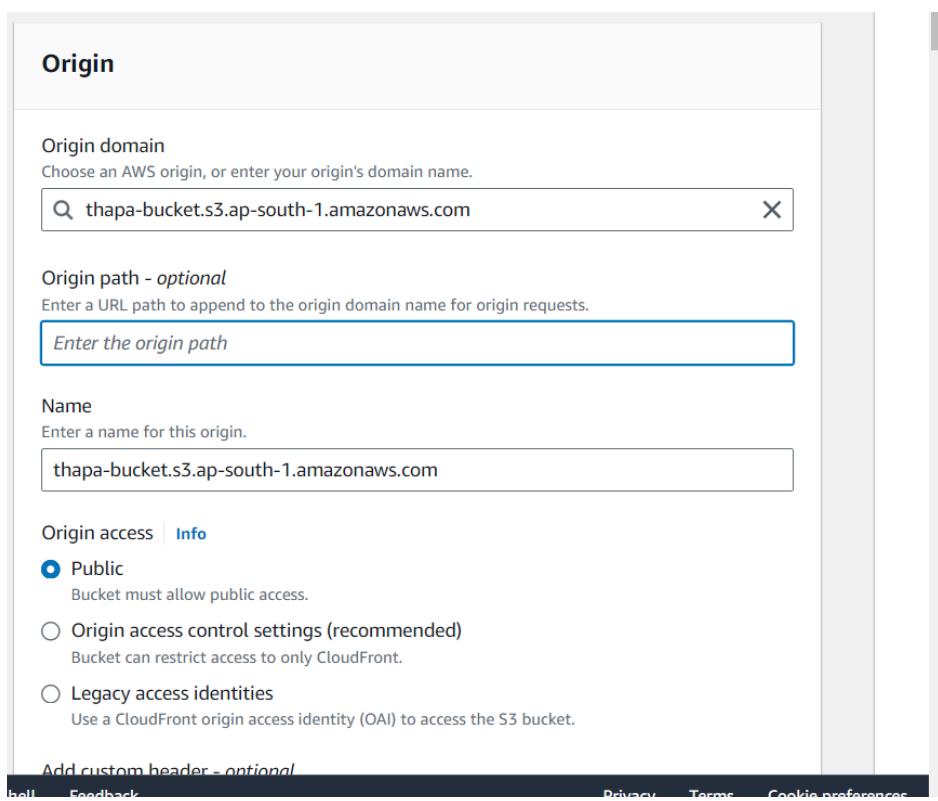
Origin Access Control (OAC) is a feature in Cloudfront, which allows you to restrict direct access to the content stored in your origin, such as an Amazon S3 bucket or a web server, ensuring that users can only access the content through the CDN distribution and not by directly accessing the origin URL.

Basically how the cloudfront can access our origin

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



A screenshot of the AWS CloudFront Distribution Details page. The top navigation bar shows 'CloudFront > Distributions > E3JCIJ900E2RYZ'. The main title is 'E3JCIJ900E2RYZ'. On the right, there is a 'View metrics' button. Below the title, there are tabs: General (selected), Security, Origins, Behaviors, Error pages, Invalidations, and Tags. The 'General' tab has a sub-section titled 'Details' containing fields for 'Distribution domain name' (dbs5cabw92oe.cloudfront.net) and 'ARN' (arn:aws:cloudfront:163679972322:distribution/E3JCIJ900E2RYZ). To the right of these fields is 'Last modified' (February 19, 2024 at 6:33:02 AM UTC). A red arrow points from the text 'dbs5cabw92oe.cloudfront.net' to the 'Distribution domain name' field. Below the 'Details' section is a 'Settings' section with fields for 'Description' (empty), 'Price class' (Set to Edge Location), 'Use all edge locations (best performance)', 'Alternate domain names' (fe.100devs.com, Custom SSL Certificate, fe.100devs.com), and 'Standard logging' (Off, Off, Off).



A screenshot of the AWS CloudFront 'Create New Origin' page. The title is 'Origin'. It has sections for 'Origin domain' (Choose an AWS origin, or enter your origin's domain name. Input field: thapa-bucket.s3.ap-south-1.amazonaws.com), 'Origin path - optional' (Enter a URL path to append to the origin domain name for origin requests. Input field: Enter the origin path), 'Name' (Enter a name for this origin. Input field: thapa-bucket.s3.ap-south-1.amazonaws.com), and 'Origin access' (Info). Under 'Origin access', the 'Public' option is selected (Bucket must allow public access.). Other options include 'Origin access control settings (recommended)' (Bucket can restrict access to only CloudFront.) and 'Legacy access identities' (Use a CloudFront origin access identity (OAI) to access the S3 bucket.). At the bottom, there is a link 'Add custom header - optional'.

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

Create new OAC

Name
The name must be unique. Valid characters: letters, numbers and most special characters. Use up to 64 characters.

Description - *optional*
The description can have up to 256 characters.

Signing behavior
 Do not sign requests
 Sign requests (recommended)
 Do not override authorization header
Do not sign if incoming request has authorization header.

Origin type

The origin type must be the same type as origin domain.

Create

⚠ You must update the S3 bucket policy

CloudFront will provide you with the policy statement after creating the distribution.

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

Default root object - *optional*

The object (file name) to return when a viewer requests the root URL (/) instead of a specific object.

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

Now we need to update S3 policy

The screenshot shows the AWS CloudFront distribution creation interface. At the top, there's a blue banner with an info icon and the text "Introducing the CloudFront Security Dashboard". Below it, a green banner says "Successfully created new distribution." A yellow warning banner at the bottom left says "The S3 bucket policy needs to be updated" with a link to "Go to S3 bucket permissions to update policy". A "Copy policy" button is also present. On the right, there are sections for "Cookie logging" (Off), "Default root object" (index.html), and "Continuous deployment" with an "Info" link. A "Create staging distribution" button is located at the bottom of the main content area.

We will copy the policy

The screenshot shows the "Bucket policy" page for an S3 bucket. It features an "Edit" and "Delete" button. A warning message in a blue-bordered box states: "Public access is blocked because Block Public Access settings are turned on for this bucket. To determine which settings are turned on, check your Block Public Access settings for this bucket. Learn more about using Amazon S3 Block Public Access." The text "using Amazon S3 Block Public Access" is a link.

Click on Edit and paste the policy

The screenshot shows the AWS S3 Bucket Policy editor. At the top, there's a search bar and a [Alt+S] keybinding indicator. On the right, there are global settings and a user dropdown. Below the header, the title "Edit bucket policy" is followed by a "Bucket policy" section. A note states: "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." Buttons for "Policy examples" and "Policy generator" are available. The "Bucket ARN" field contains "arn:aws:s3:::thapa-bucket". The "Policy" section displays the following JSON code:

```
1 {  
2     "Version": "2008-10-17",  
3     "Id": "PolicyForCloudFrontPrivateContent",  
4     "Statement": [  
5         {  
6             "Sid": "AllowCloudFrontServicePrincipal",  
7             "Effect": "Allow",  
8             "Principal": {  
9                 "Service": "cloudfront.amazonaws.com"  
10            },  
11            "Action": "s3:GetObject",  
12            "Resource": "arn:aws:s3:::thapa-bucket/*",  
13            "Condition": {  
14                "StringEquals": {  
15                    "AWS:SourceArn": "arn:aws:cloudfront::975049934835:distribution/E2P3VXNI6NT7W4"  
16                }  
17            }  
18        }  
19    ]  
20}
```

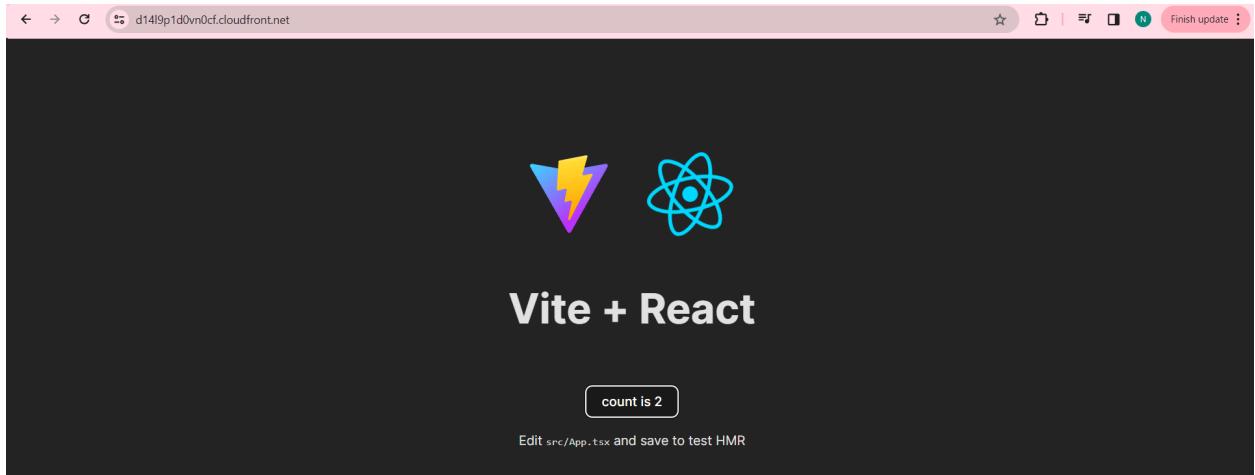
To the right of the policy code, there's an "Edit statement" button and a "Select a statement" panel with a "Add new statement" button.

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

The screenshot shows the AWS CloudFront Distributions details page for distribution ID "E2P3VXNI6NT7W4". The top navigation bar includes "Services", a search bar, and a user dropdown. The main content area shows the distribution name "E2P3VXNI6NT7W4" and a "View metrics" button. Below the name, there are tabs for "General", "Security", "Origins", "Behaviors", and "Error pages". The "General" tab is selected. The "Details" section contains the following information:

Distribution domain name	ARN
d14l9p1d0vn0cf.cloudfront.net	arn:aws:cloudfront::975049934835:distribution/E2P3VXNI6NT7W4
Last modified	March 30, 2024 at 9:06:51 AM UTC

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

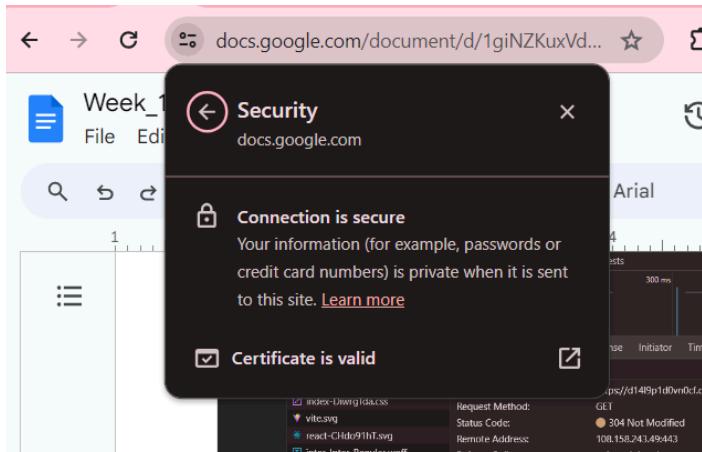
Name	Headers	Preview	Response	Initiator	Timing
d14l9p1d0vn0cf.cloudfront.net	General				
index-MJNRYYu.js					
index-DiwrqTda.css					
vite.svg					
react-CHdo91hT.svg					
inter_Inter-Regular.woff					
inter_Inter-Medium.woff					
inter_Inter-Bold.woff					
vite.svg	Response Headers				

Connect your 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