

# Step 1 - How to deploy Frontends to AWS



- New things we will learn include
- 1. Object stores (S3)
- 2. CDNs (Cloudfront)

Step 1 - Signup and get an AWS account.

Step 2 - Make sure you can access S3 and cloudfront (this will automatically happen if you are the root user of that account)

The screenshot shows the AWS Console Home page. On the left, under 'Recently visited' services, there is a list including Route 53, AWS FIS, EC2, Simple Queue Service, IAM, CloudFront, Elastic Container Service, Athena, and EFS. Two red arrows point from the S3 and CloudFront icons in this list to their respective sections on the right. On the right, the 'Applications' section shows '0' applications, with a 'Create application' button. The region is set to 'Asia Pacific (Mumbai)' and the current region is 'ap-south-1'. A search bar for 'Find applications' is also present.

# Step 2 - 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

## Go to your react project

```
cd /link/to/your/react/proj...Copy
```

## Build your project

```
npm run bu...Copy
```

## Try serving the HTML/CSS/JS locally

```
npm i -g serveCopy
```

At this point you have basic HTML/CSS/JS code that you can deploy on the internet.

You might be tempted to host this on an EC2 instance, but that is not the right approach

The next slide explains why

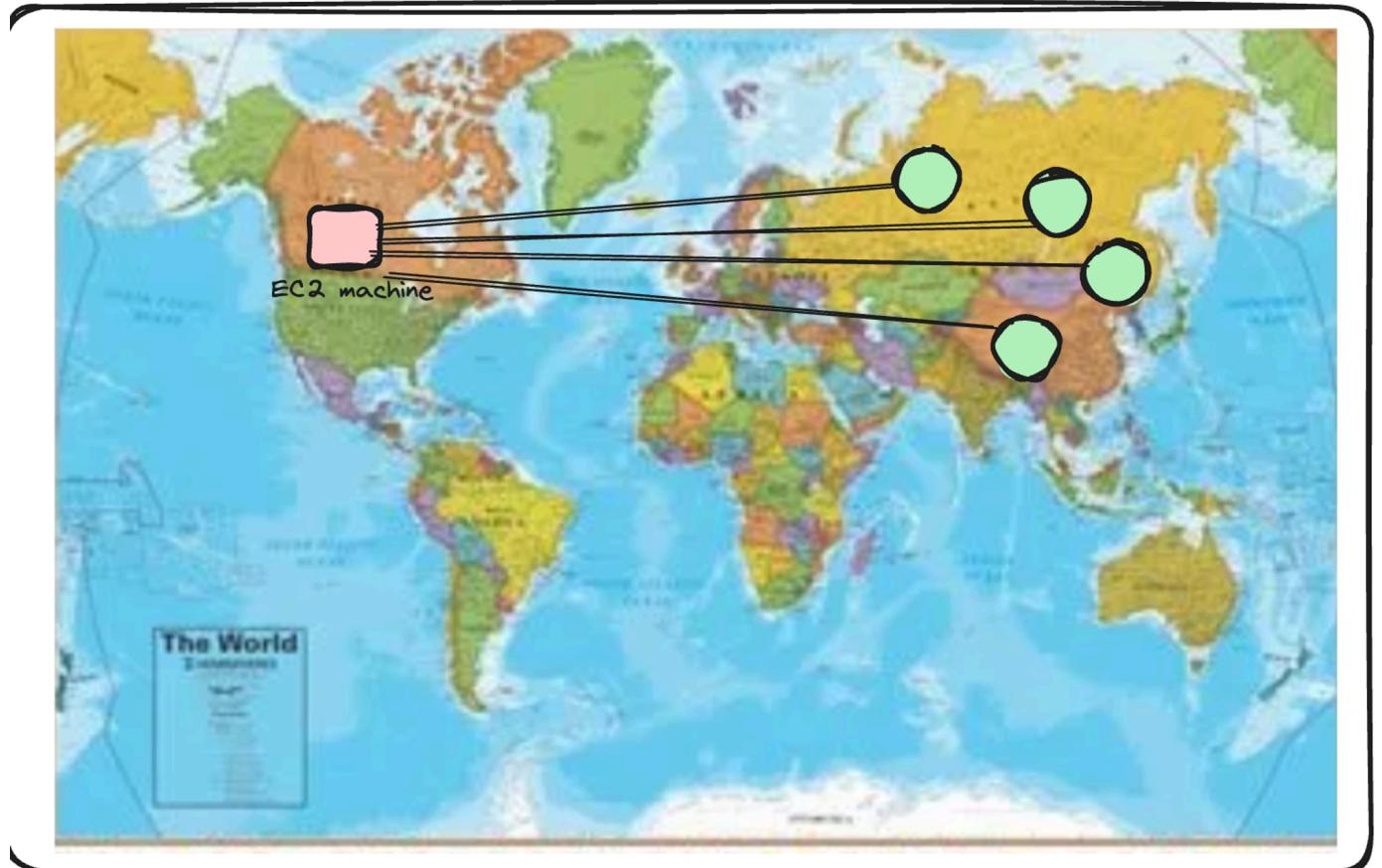
# Step 3 - What are CDNs?

A CDN stands for **Content Delivery Network**.

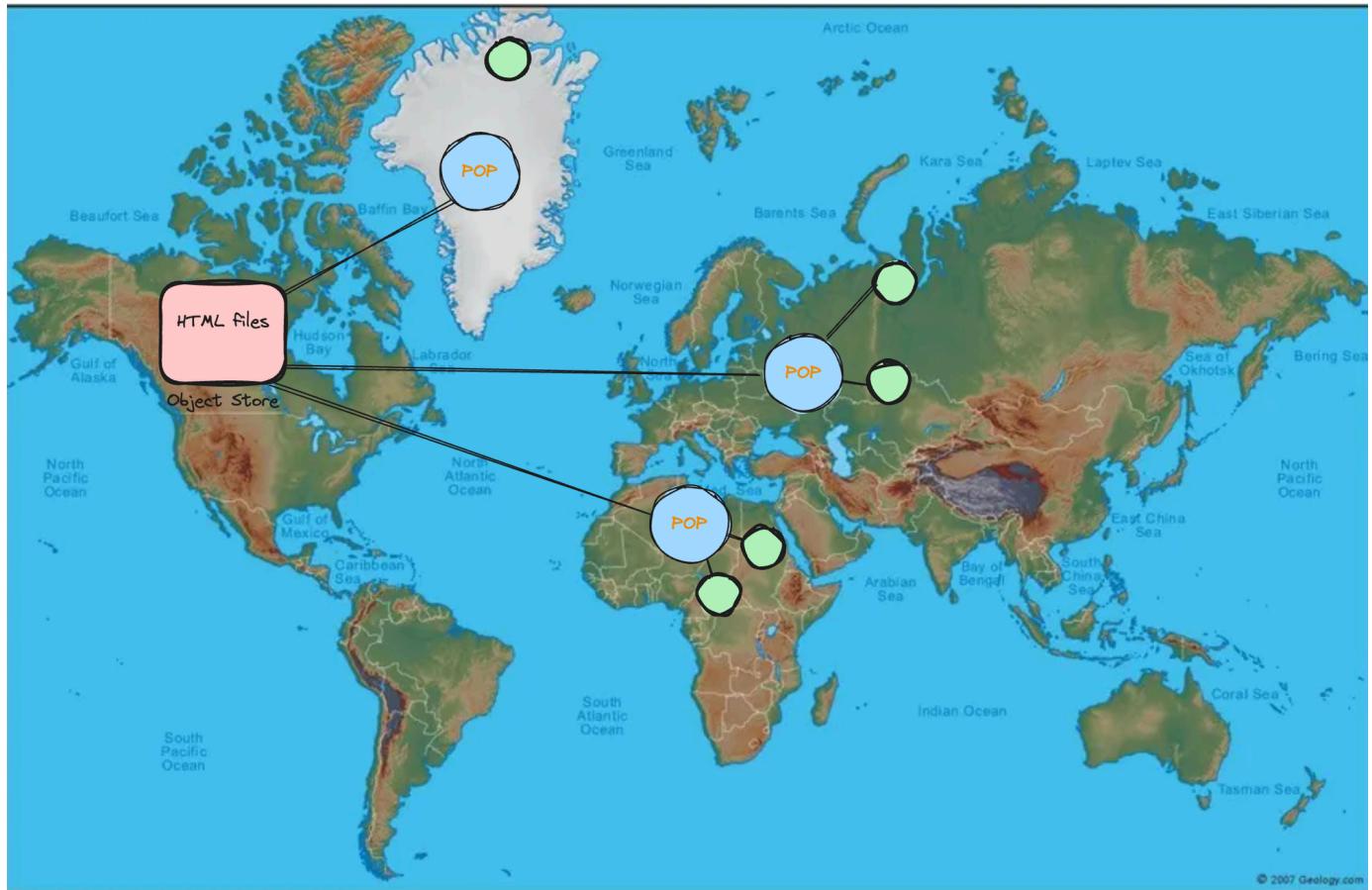
As the name suggests, it's an optimal way for you to deliver content (mp4 files, jpgs and even HTML/CSS/JS files) to your users.

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

## 1. EC2 machine approach



## 2. CDN approach



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.

Great video on how Hotstar scales their infrastructure during cricket matches (they use CDNs heavily)



# Step 4 - 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 summary statistics: 2.4 PB, 2.5 G, 1.2 MB, 420.9 K, 11.4 GB, and 255.0 GB. Below this, there are two tabs: 'General purpose buckets' (selected) and 'Directory buckets'. A search bar labeled 'Find buckets by name' is present. The main table lists one bucket: 'test11123123' located in 'Asia Pacific (Mumbai) ap-south-1'. The table has columns for Name, AWS Region, Access, and Creation date. At the top right of the table, there are buttons for 'Create', 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'. A red arrow points from the text above to the 'Create bucket' button.

The screenshot shows the 'Create bucket' wizard. The first step, 'General configuration', is selected. It includes fields for 'AWS Region' (set to 'Asia Pacific (Mumbai) ap-south-1'), 'Bucket name' (set to 'kirat-test'), and 'Copy settings from existing bucket - optional' (with a 'Choose bucket' button). A note states that the bucket name must be unique and follow naming rules. The second step, 'Object Ownership', is partially visible at the bottom.

The screenshot shows the AWS S3 console. At the top, a green banner indicates that a bucket named "kirat-test" has been successfully created. Below the banner, the "Buckets" section is visible, showing an "Account snapshot" with last updated information and a "View Storage Lens dashboard" button. The main area displays the contents of the "dist" folder within the "kirat-test" bucket, which contains three files: "assets", "index.html", and "vite.svg".

# Step 5 - Upload the file bundle to S3

Upload all the files in the `dist` folder of your react project to S3

The screenshot shows the AWS S3 console with a red arrow pointing from the "Upload" button at the bottom of the left sidebar to the "Upload" icon inside the "dist" folder view on the right. This visual cue guides the user to click the "Upload" button to begin the process of uploading the files from the local "dist" folder to the S3 bucket.

# Step 6 - Try accessing the website

The screenshot shows a web browser window with the following details:

- Address bar: kirat-test-2.s3.eu-west-3.amazonaws.com/index.html
- Tab bar: https://getsubmiss..., hotel in paris - Go..., ad, New Tab
- Content area:
  - A message stating: "This XML file does not appear to have any style information associated with it. The document tree is shown below."
  - An XML error response:

```
<Error>
  <Code>AccessDenied</Code>
  <Message>Access Denied</Message>
  <RequestId>T3YE0RGZZXH90J2W</RequestId>
  <HostId>HGnifF+p0hpzxbx3fll99MwqAQN5Y5bVN7hmLexMojiy9bk1wUCKqXDWUm3tuXCqI9fkli3jMQ=</HostId>
</Error>
```

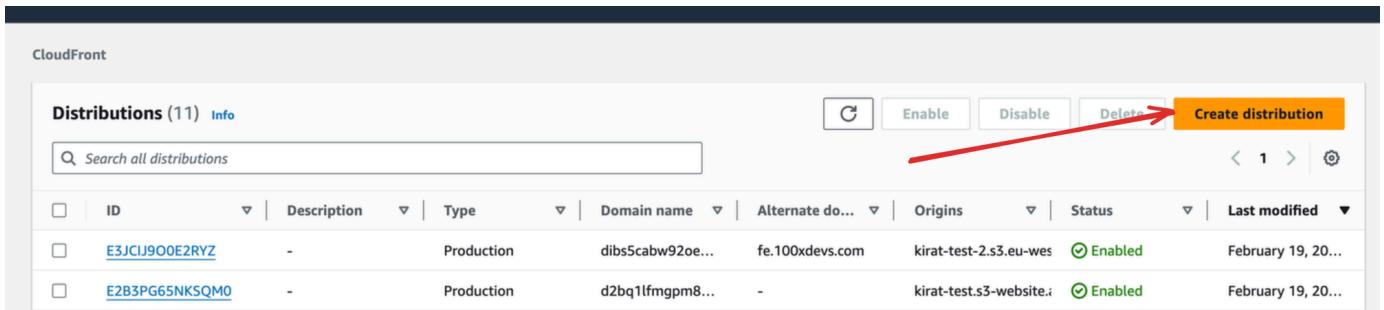
You might be tempted to open your S3 bucket at this point, but don't

Your S3 bucket should be blocked by default, and you should allow CloudFront (CDN) to access it.

# Step 7 - Connecting Cloudfront

## Step 1 - Create cloudfront distribution

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



The screenshot shows the AWS CloudFront 'Distributions' page. At the top, there's a search bar labeled 'Search all distributions'. Below it is a table with columns: ID, Description, Type, Domain name, Alternate do..., Origins, Status, and Last modified. Two distributions are listed:

ID	Description	Type	Domain name	Alternate do...	Origins	Status	Last modified
E3JCIJ900E2RYZ	-	Production	dibs5cabw92oe...	fe.100xdevs.com	kirat-test-2.s3.eu-wes	Enabled	February 19, 20...
E2B3PG65NKSQMO	-	Production	d2bq1lmgpm8...	-	kirat-test.s3-website.i...	Enabled	February 19, 20...

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

## Origin

### Origin domain

Choose an AWS origin, or enter your origin's domain name.



### Origin path - *optional*

Enter a URL path to append to the origin domain name for origin requests.

### Name

Enter a name for this origin.

### Origin access | [Info](#)

#### Public

Bucket must allow public access.

#### Origin access control settings (recommended)

Bucket can restrict access to only CloudFront.

#### Legacy access identities

Use a CloudFront origin access identity (OAI) to access the S3 bucket.

#### Origin access control

Select an existing origin access control (recommended) or create a new control.



⚠ This field cannot be empty

⚠ **You must update the S3 bucket policy**

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.

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

# Step 8 - Connect your own domain to it

Websites aren't fun if you have to go to a URL that looks like this -

<https://dibs5cabw92oe.cloudfront.net>

Connect your own custom domain by following the given steps -

## 1. Select edit on the root page

The screenshot shows the AWS CloudFront Distribution settings page. The distribution name is E2B3PG65NKSQMO. The 'General' tab is selected. In the 'Settings' section, there is a row for 'Alternate domain names' which is currently empty. To the right of this row is a column with logging and root object settings. At the bottom right of the 'Settings' section, there is an 'Edit' button, which is highlighted with a red arrow.

## 2. Attach a domain name to the distribution

## Edit settings

### Settings

#### Price class | [Info](#)

Choose the price class associated with the maximum price that you want to pay.

- Use all edge locations (best performance)
- Use only North America and Europe
- Use North America, Europe, Asia, Middle East, and Africa

#### Alternate domain name (CNAME) - *optional*

Add the custom domain names that you use in URLs for the files served by this distribution.

[Remove](#)[Add item](#)

 To add a list of alternative domain names, use the [bulk editor](#).

#### Custom SSL certificate - *optional*

Associate a certificate from AWS Certificate Manager. The certificate must be in the US East (N. Virginia) Region (us-east-1).

[Request certificate](#)

#### Supported HTTP versions

Add support for additional HTTP versions. HTTP/1.0 and HTTP/1.1 are supported by default.

- HTTP/2
- HTTP/3

#### Default root object - *optional*

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

## 3. Create a certificate

Since we want our website to be hosted on HTTPS, we should request a certificate for our domain

## Edit settings

### Settings

#### Price class [Info](#)

Choose the price class associated with the maximum price that you want to pay.

- Use all edge locations (best performance)
- Use only North America and Europe
- Use North America, Europe, Asia, Middle East, and Africa

#### Alternate domain name (CNAME) - *optional*

Add the custom domain names that you use in URLs for the files served by this distribution.

[Remove](#)[Add item](#)

 To add a list of alternative domain names, use the [bulk editor](#).

#### Custom SSL certificate - *optional*

Associate a certificate from AWS Certificate Manager. The certificate must be in the US East (N. Virginia) Region (us-east-1).

[Request certificate](#) 

#### Supported HTTP versions

## Step 4 - Follow steps to create the certificate in the certificate manager



## Step 9 - Error pages

You will notice a problem, whenever you try to access a route on your page that isn't the index route (/user/1) , you reach an error page

## 403 Forbidden

- Code: AccessDenied
- Message: Access Denied
- RequestId: V17AB7NEC9FRRDWX
- HostId: OJzE4K3MrightV9+NivXXYCb1ueDb26lEZ6MEVL99vUfhQZkiYXW9K1IUjtAvpFMRyx/IoMVnqaw=

This is because CloudFront is looking for a file `/user/1` in your S3, which doesn't exist.

To make sure that all requests reach `index.html`, add an `error page` that points to `index.html`

HTTP error code	▲   Minimum TTL (seconds)	▼   Response page path	▼   HTTP response code
403	0	index.html	200

## Edit custom error response

### Error response Info

#### HTTP error code

Customize the custom error response when the origin sends this error code.

404: Not Found ▾

#### Error caching minimum TTI



You might have to invalidate cache to see this in action.