

# DailyCode

↗



## Frontend Deployments on AWS (1 / 9)

# 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', there are links to Route 53, EC2, Elastic Kubernetes Service, IAM, Elastic Container Service, Athena, and EFS. To the right, under 'Applications (0)', it says 'Region: Asia Pacific (Mumbai)' and lists 'ap-south-1 (Current Region)'. A search bar 'Find applications' is present. Below the search bar is a table with columns 'Name', 'Description', 'Region', and 'Originating account'. The table is empty, showing 'No applications'. A 'Create application' button is at the bottom. Red arrows point from the 'S3' and 'CloudFront' icons in the 'Recently visited' list to their corresponding rows in the 'Applications' table.





# DailyCode



Frontend Deployments on AWS (2 / 9)

## 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/project



### Build your project

npm run build



### Try serving the HTML/CSS/JS locally

npm i -g serve  
serve



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





# DailyCode



Frontend Deployments on AWS (3 / 9)

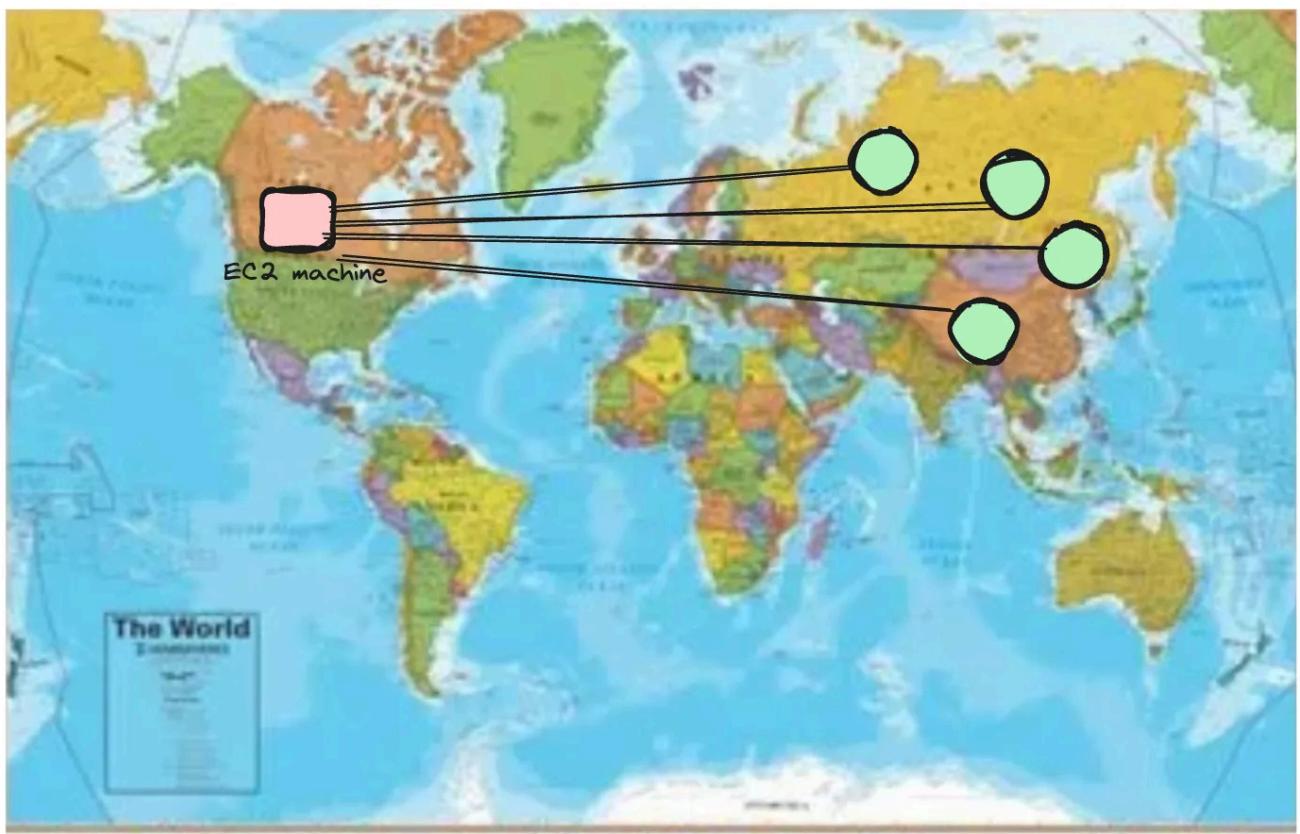
## Step 3 – What are CDNs?

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

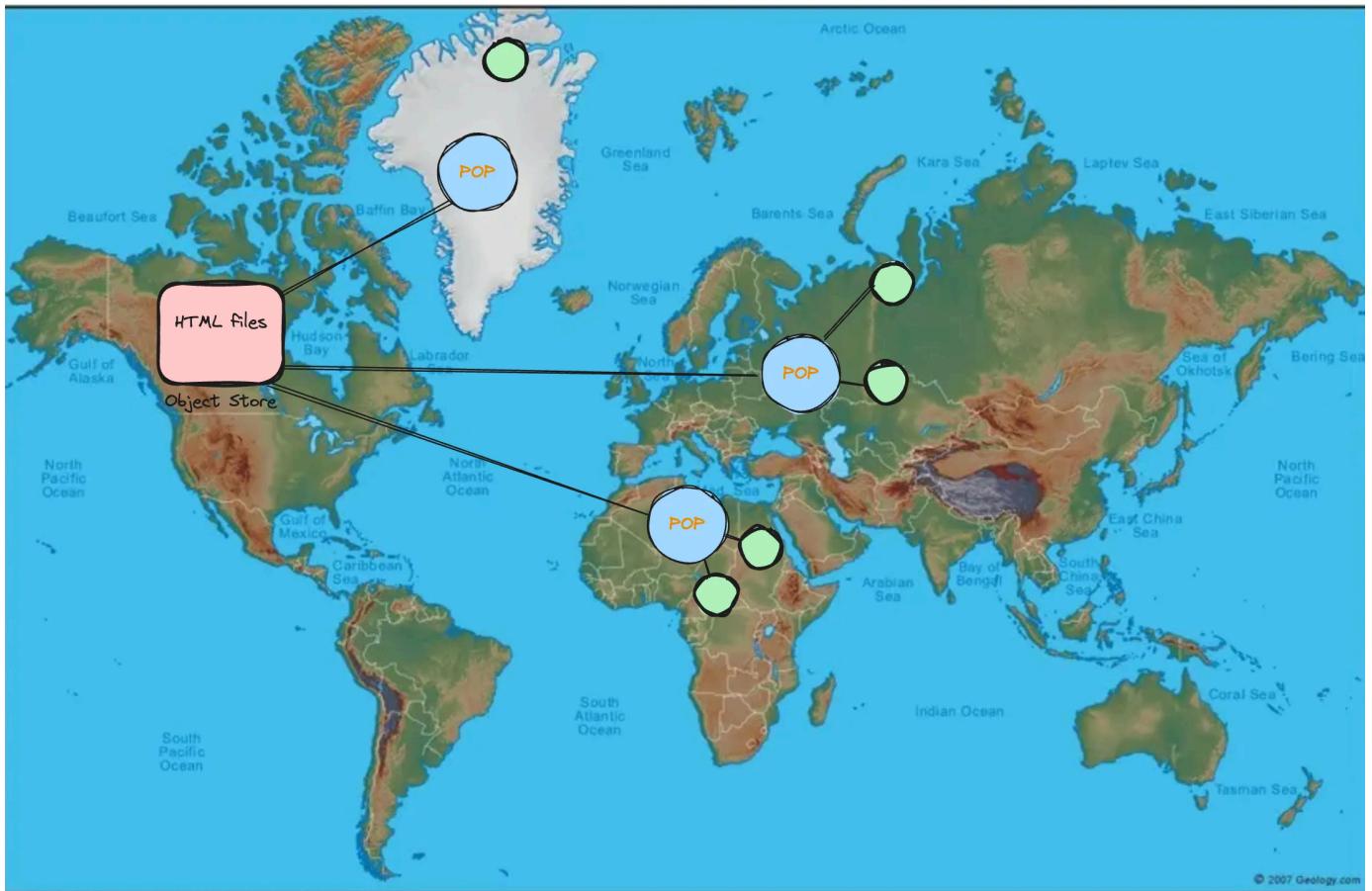
As the name suggests, it's an optimal way for you to deliver content (mp4 files, jpgs and even HTML/CSS/JS files) to your users.

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)

# How Hotstar Scaled 25 Million Users

Concurrency Pattern!

25.3M

18.2M

13.9M

Day One

Day Two

Day Three

Start

Mid

End

200K+ VIEWS

hotstar tech\_





# DailyCode



Frontend Deployments on AWS (4 / 9)

## 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 six small icons: a left arrow, an empty box, an empty box, a sun, an empty box, and a profile picture. Below the header, the text "Frontend Deployments on AWS (4 / 9)" is displayed. The main content area is titled "Step 4 – Creating an object store in AWS". A sub-section states: "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." Below this text is a screenshot of the AWS S3 console. The screenshot shows a list of "General purpose buckets" with 12 items. The first item is "test11123123". The interface includes a search bar, filter dropdowns for "Name", "AWS Region", "Access", and "Creation date", and buttons for "Copy ARN", "Empty", "Delete", and "Create bucket". An orange arrow points from the text above to the "Create bucket" button.

[Amazon S3](#) > [Buckets](#) > Create bucket

## Create bucket Info

Buckets are containers for data stored in S3. [Learn more](#)

### General configuration

#### AWS Region

Asia Pacific (Mumbai) ap-south-1

#### Bucket name Info

kirat-test

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

#### Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Format: s3://bucket/prefix

### Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership

The screenshot shows a green success notification bar at the top of the AWS S3 interface. The message reads: "Successfully created bucket 'kirat-test'. To upload files and folders, or to configure additional bucket settings, choose View details." Below this, the standard S3 navigation bar is visible, showing "Amazon S3 > Buckets". A "View Storage Lens dashboard" button is also present. The main content area is currently empty, showing a placeholder for account snapshot information.



# DailyCode

↗



## Frontend Deployments on AWS (5 / 9)

# Step 5 – Upload the file bundle to S3

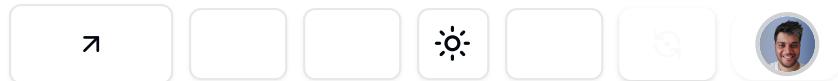
Upload all the files in the `dist` folder of your react project to S3

The screenshot shows the AWS S3 console interface. On the left, the navigation bar shows 'Amazon S3 > Buckets > kirat-test'. The main area displays the contents of the 'dist' folder. Inside the folder, there are three items: a folder named 'assets', a file named 'index.html', and a file named 'vite.svg'. A red box highlights these three items. At the bottom left of the S3 interface, there is an 'Upload' button. A red arrow points from this 'Upload' button towards the highlighted files in the 'dist' folder.





# DailyCode



Frontend Deployments on AWS (6 / 9)

## Step 6 – Try accessing the website

A screenshot of a web browser window. The address bar shows the URL `kirat-test-2.s3.eu-west-3.amazonaws.com/index.html`. Below the address bar, there are several browser tabs and icons. A message in the main content area reads: "This XML file does not appear to have any style information associated with it. The document tree is shown below." Below this message, an XML error response is displayed:

```
<Error>
  <Code>AccessDenied</Code>
  <Message>Access Denied</Message>
  <RequestId>T3YE0RGZZXH00J2W</RequestId>
  <HostId>HGnifF+p0hpzxbx3fll99MWqAQn5Y5bVVN7hmLexMojiy9bk1wUCKqXDWUm3tuXCqI9fkli3jMQ=</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.





# DailyCode



Frontend Deployments on AWS (7 / 9)

## 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 header with the CloudFront logo and a search bar labeled "Search all distributions". Below the header 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	d2bq1lfgmpm8...	-	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 area.

### 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.



CloudFront > Distributions > E3JCIJ9O0E2RYZ

## E3JCIJ9O0E2RYZ

[View metrics](#)

General Security Origins Behaviors Error pages Invalidations Tags

### Details

Distribution domain name <a href="#">dibs5cabw92oe.cloudfront.net</a>	ARN <a href="#">arn:aws:cloudfront::163679972322:distribution/E3JCIJ9O0E2RYZ</a>	Last modified February 19, 2024 at 6:33:02 AM UTC
--	---	--

### Settings

Description -

Price class -

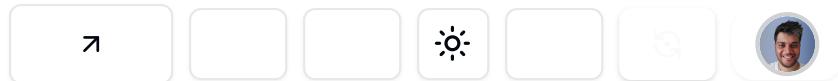
Use all edge locations (best performance)  
Supported HTTPD version

Alternate domain names <a href="#">fe.100xdevs.com</a> Custom SSL certificate <a href="#">fe.100xdevs.com</a>	Standard logging Off Cookie logging Off <small>Default to next object</small>
--	---

[Edit](#)



# DailyCode



## Frontend Deployments on AWS (8 / 9)

# 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 top navigation bar includes 'CloudFront' > 'Distributions' > 'E2B3PG65NKSQMO'. The main content area has tabs for 'General', 'Security', 'Origins', 'Behaviors', 'Error pages', 'Invalidations', and 'Tags'. The 'General' tab is selected. Under the 'Details' section, the 'Distribution domain name' is listed as 'd2bq1lfpngpm8dp.cloudfront.net'. The 'ARN' field shows 'arn:aws:cloudfront::163679972322:distribution/E2B3PG65NKSQMO'. The 'Last modified' field shows 'February 19, 2024 at 5:01:19 AM UTC'. In the 'Settings' section, there are fields for 'Description' (empty), 'Price class' (empty), 'Alternate domain names' (empty), and a table for 'Standard logging', 'Cookie logging', and 'Default root object' (all empty). A red arrow points to the 'Edit' button in the 'Settings' section.

## 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

Certificate status					
Identifier fb2088a7-c75f-4a42-91af-2a7fb4600a0e	Status Issued				
ARN <a href="#">arn:aws:acm:us-east-1:163679972322:certificate/fb2088a7-c75f-4a42-91af-2a7fb4600a0e</a>	Type Amazon Issued				
Domains (1)					
<a href="#">Create records in Route 53</a> <a href="#">Export to CSV </a> <span style="float: right;">&lt; 1 &gt;</span>					
Domain	Status	Renewal status	Type	CNAME name	CNAME value
fe.100xdevs.com	Success	-	CNAME	_a96f3ef5c0e0ef282152985dfb428092.fe.100xdevs.com.	_21845a5bfaa0b0cbbb6b8a55b28c5501.mhbtsbpndt.acm-validations.aws.
Details					
In use	Serial number	Requested at	Renewal eligibility		

 These DNS settings are active. Changes are published immediately, but may take time to propagate

## Resource records

 Export DNS records

Resource records point to the services that your domain uses, including web and email services. [Learn more about resource records](#)

### Custom records

100xdevs.com/A, \_a96f3ef5c0e0ef282152985dfb428092.fe.100xdevs.com/CNAME and 27 more

[Manage custom records](#)

Host name	Type	TTL	Data
100xdevs.com	A	1 hour	76.76.21.21
_a96f3ef5c0e0ef282152985dfb428092.f e.100xdevs.com	CNAME	1 hour	_21845a5bfaa0b0cbbb6b8a55b28c5501.mhbtsbpdnt.acm- validations.aws.



## Step 5 – Add a CNAME record for the website to point to your cloudfront URL

fe.100xdevs.com      CNAME      1 hour      dibs5cabw92oe.cloudfront.net.

That's it, you have a fully running react project hosted on HTTPS on a custom domain



# DailyCode

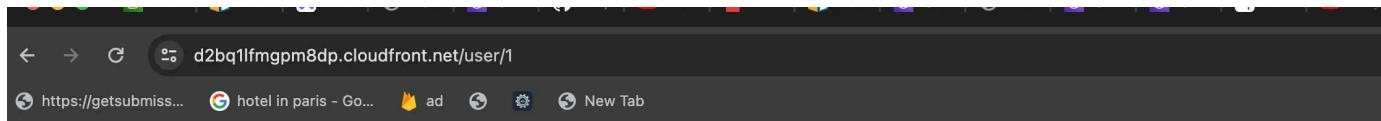
↗



Frontend Deployments on AWS (9 / 9)

## 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: OJzE4K3MrlighV9+NivXXYCb1ueDb26IEZ6MEVL99vUfhQZkiYXW9K1IUjtAvpFMRyx/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 TTL

Enter the error caching minimum time to live (TTL), in seconds.

0

#### Customize error response

Send a custom error response instead of the error received from the origin.

- No  
 Yes

#### Response page path

Enter the path to the custom error response page.

/index.html

#### HTTP Response code

Choose the HTTP status code to return to the viewer. CloudFront can return a different status code to the viewer than what it received from the origin.

200: OK ▾

Cancel

Save changes



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



