Agenda

- (Background) Bundling a Web app.

- Frontend app deployment to AWS platform.
  - What services?

- Full-stack web app.
  - Deployment as a single CDK app.
  - Custom domain name.

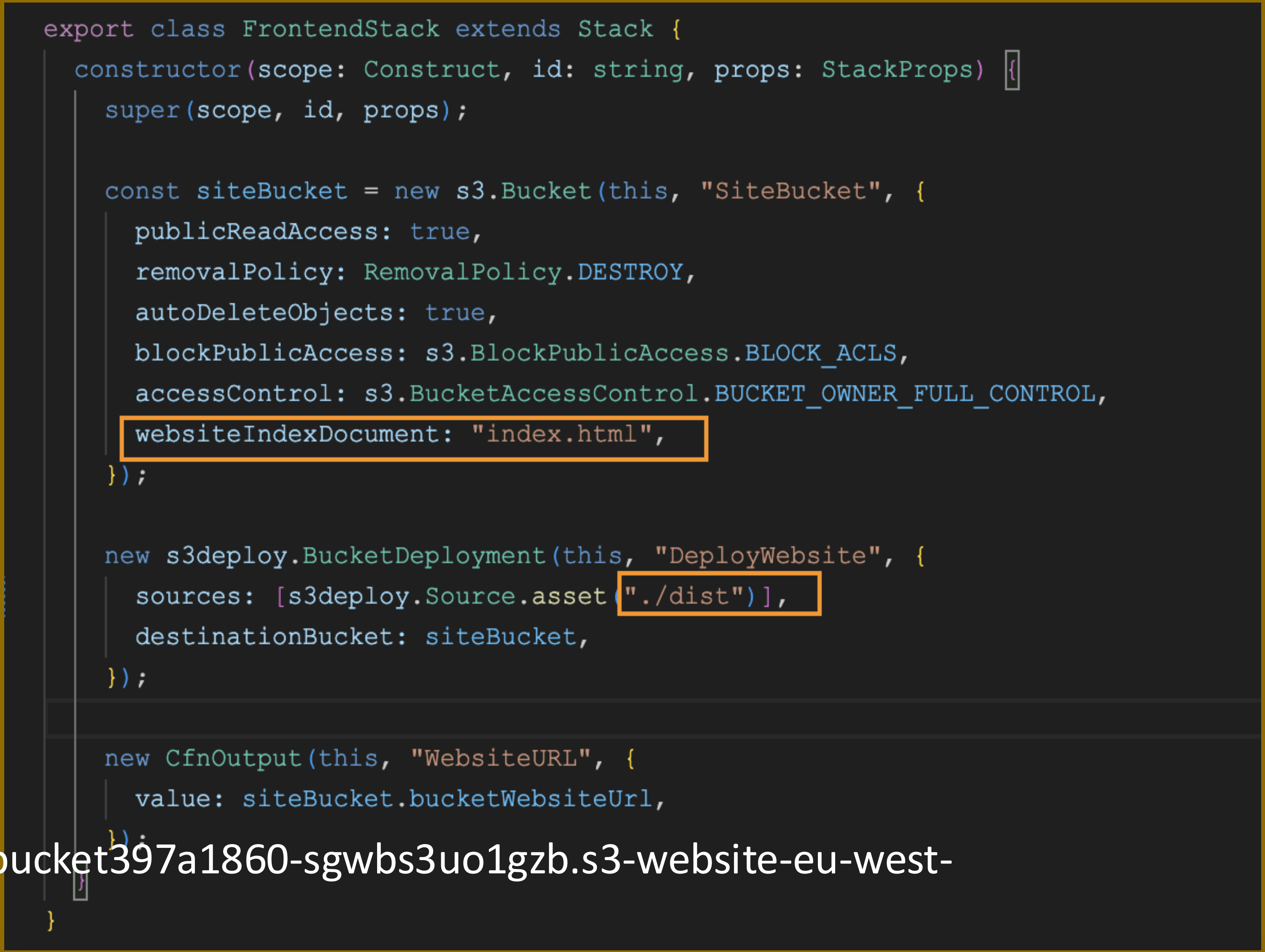
Web App bundling

- Development versus Production environment.
  - Hot module replacement (HMR)
- Dynamic versus static version of a web app

```
$ cd myWebApp
$ npm run dev      (Start development server; HMR support)
$ npm i serve –g   (Simple web server ; No HMR support)
$ npm run build    (Build static version of app; BUNDLING)
$ serve ./dist     (Host static web app / website)
```

S3 Website hosting

- We can configure an S3 bucket to host a website or static web app.



```
export class FrontendStack extends Stack {
  constructor(scope: Construct, id: string, props: StackProps) {
    super(scope, id, props);

    const siteBucket = new s3.Bucket(this, "SiteBucket", {
      publicReadAccess: true,
      removalPolicy: RemovalPolicy.DESTROY,
      autoDeleteObjects: true,
      blockPublicAccess: s3.BlockPublicAccess.BLOCK_ACLS,
      accessControl: s3.BucketAccessControl.BUCKET_OWNER_FULL_CONTROL,
      websiteIndexDocument: "index.html",
    });

    new s3deploy.BucketDeployment(this, "DeployWebsite", {
      sources: [s3deploy.Source.asset("./dist")],
      destinationBucket: siteBucket,
    });

    new CfnOutput(this, "WebsiteURL", {
      value: siteBucket.bucketWebsiteUrl,
    });
  }
}
```

http://s3website-sitebucket397a1860-sgwbs3uo1gzb.s3-website-eu-west-1.amazonaws.com/

AWS CloudFront (CDN)

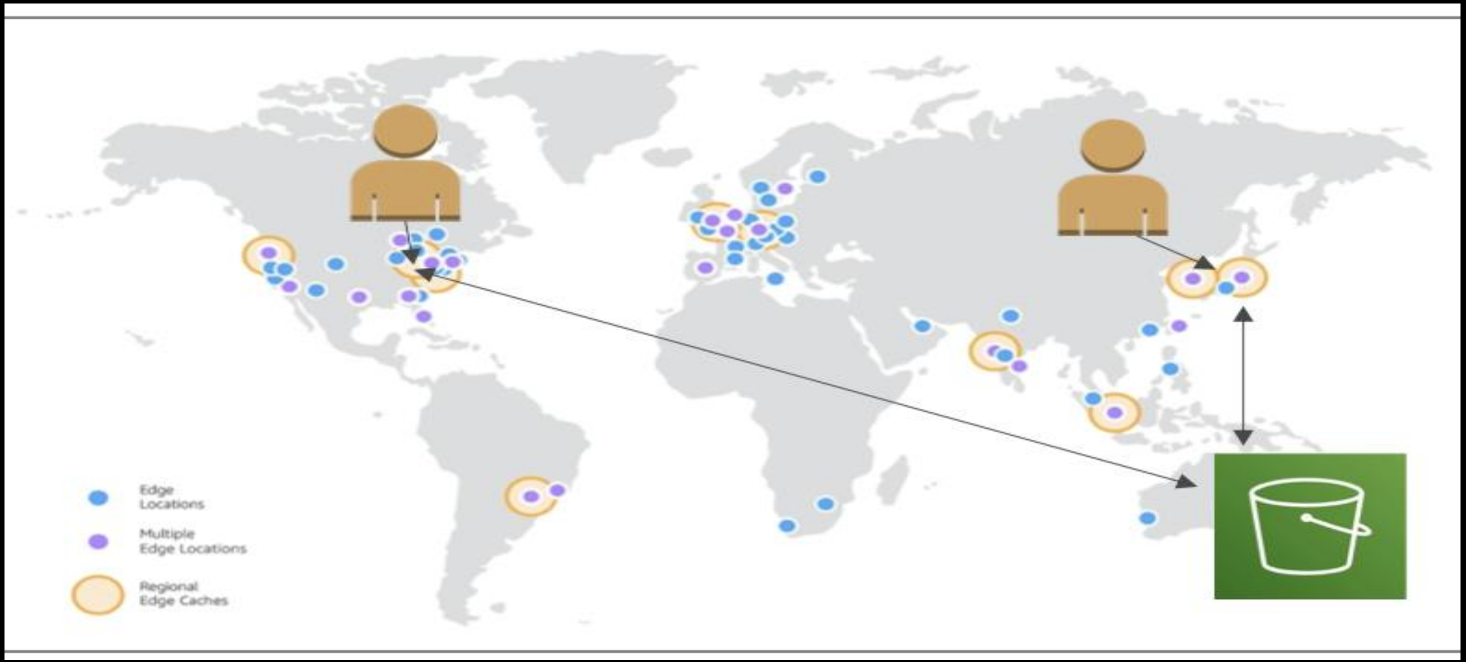
CloudFront / CDN

- A CDN (Content Delivery Network) is a network of geographically distributed servers that <u>cache</u> copies of content close to the end users, thus lowering <u>latency</u> when they download or stream content.
  - Servers = Edge locations or Points of Presence (POPs).
  - Content = Web assets, Documents, Videos, etc
- CloudFront – 400+ POPs in 90 cities and across 47 different countries.
- DDoS protection.

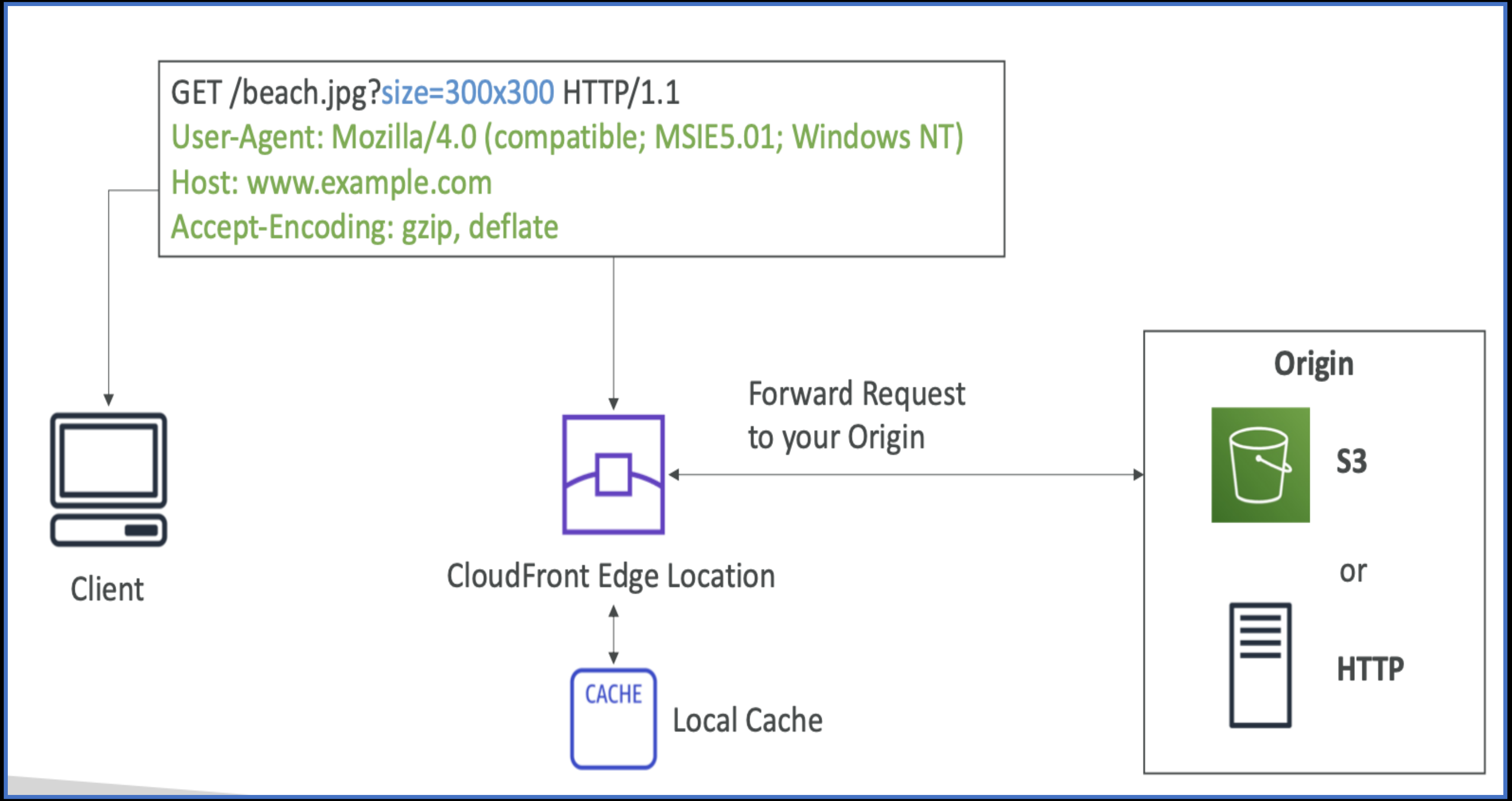
ClousFront - Resources

- Two key infrastructure resource types:

1. Origin – The source of your content, e.g. S3.
2. Distribution - A distribution tells CloudFront where you want content to be delivered from, and the details about how to track and manage content delivery.



CloudFront resources


```
GET /beach.jpg?size=300x300 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.example.com
Accept-Encoding: gzip, deflate
```

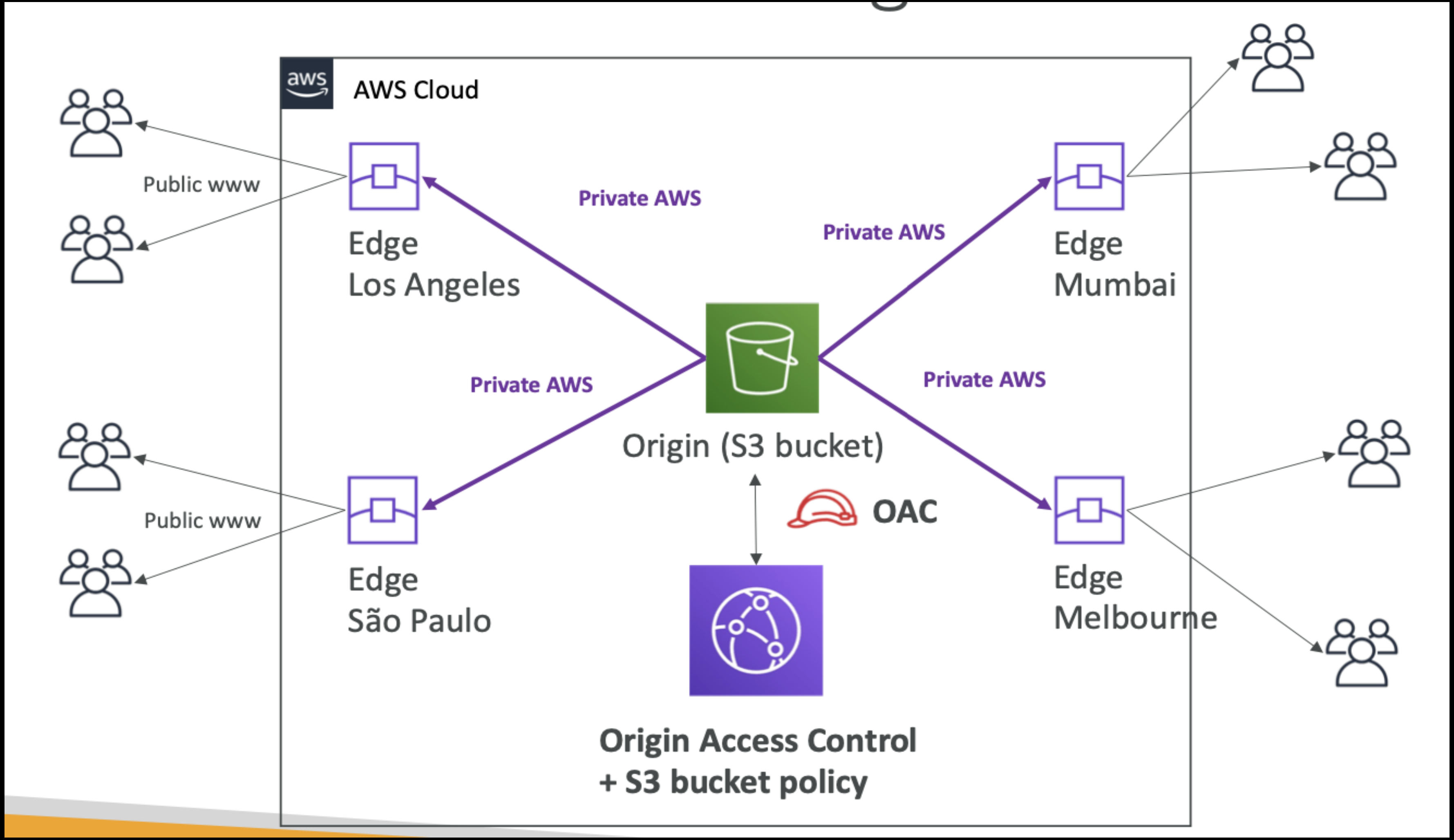
Cloudfront Origins.

- Source of your content.

1. S3 bucket.
   - Enhanced security with CloudFront Origin Access Identity (OAI)
     - Origin Access Control (OAC) is replacing OAI.
   - CloudFront can also be used as an ingress (to upload files to S3).

2. Custom origin (HTTP).
   - API Gateway.
   - Application load Balancer.
   - EC2 instance.
   - S3 website.

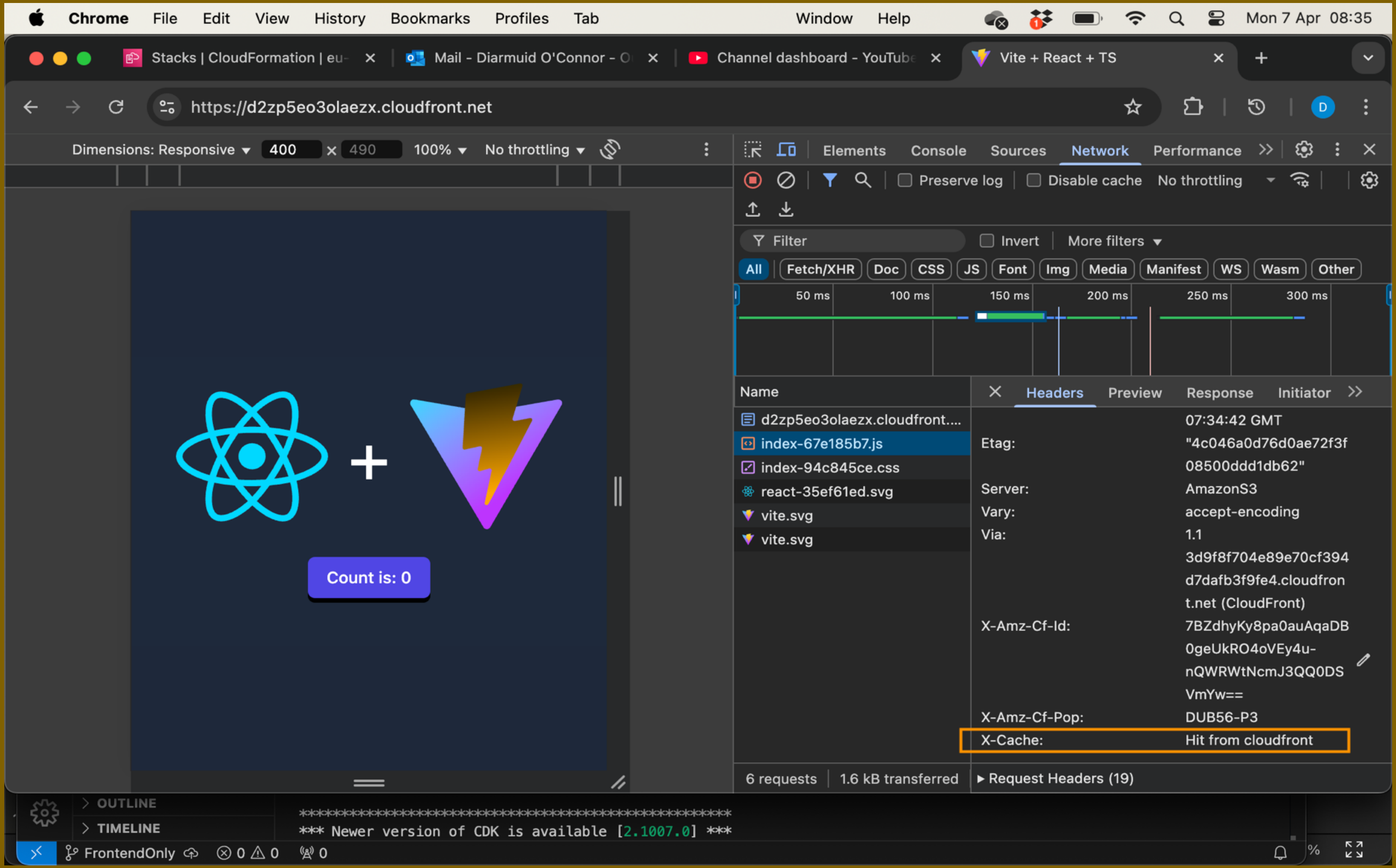
S3 as a Cloudfront Origin.



Demo

- Objective: Provision a CloudFront distribution for a standalone React app, using S3 as the origin.

Demo



https://d2zp5eo3olaezx.cloudfront.net

Name
d2zp5eo3olaezx.cloudfront....
index-67e185b7.js
index-94c845ce.css
react-35ef61ed.svg
vite.svg
vite.svg

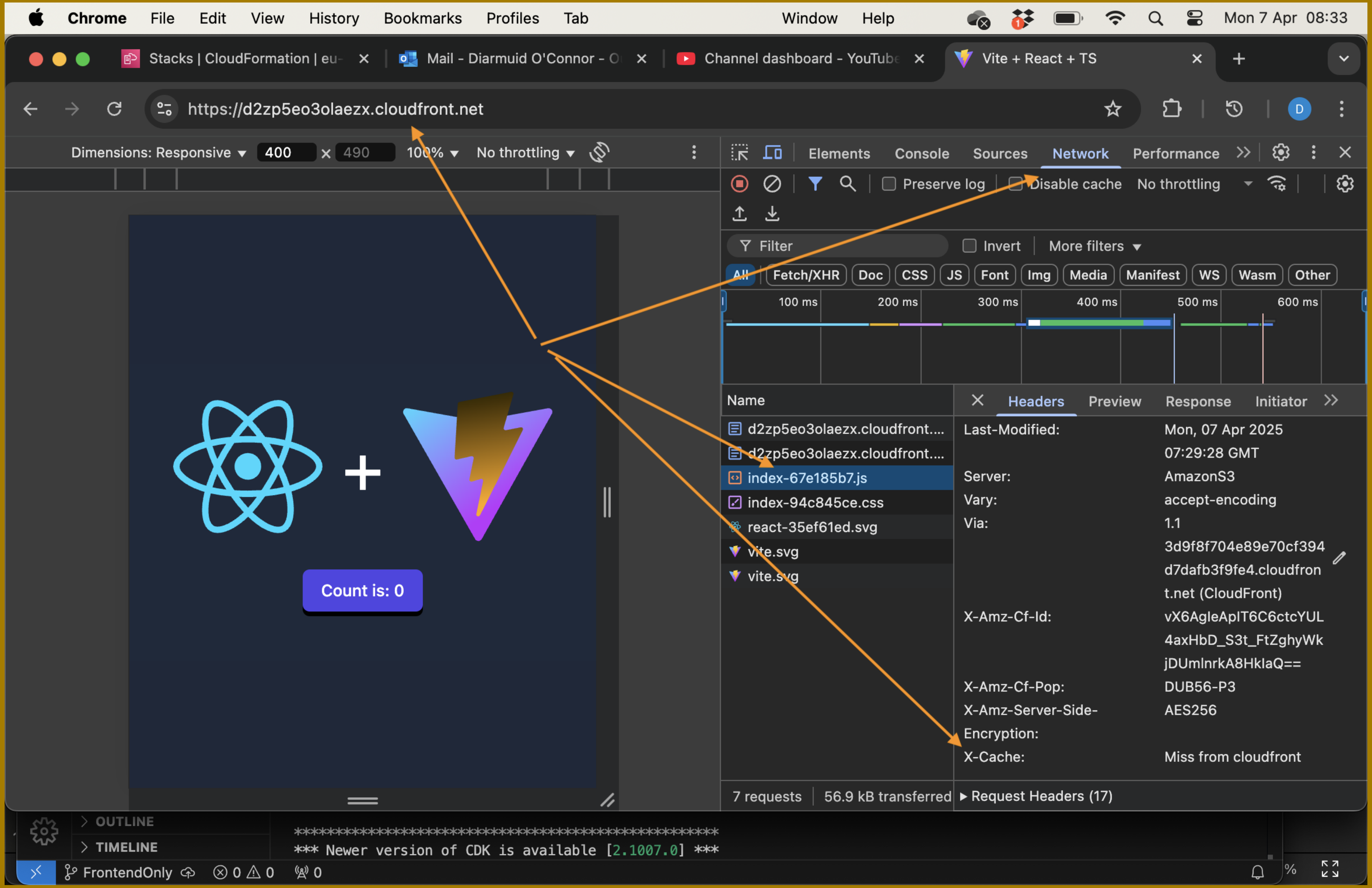
6 requests | 1.6 kB transferred

	07:34:42 GMT
Etag:	"4c046a0d76d0ae72f3f08500ddd1db62"
Server:	AmazonS3
Vary:	accept-encoding
Via:	1.1 3d9f8f704e89e70cf394d7dafb3f9fe4.cloudfront.net (CloudFront)
X-Amz-Cf-Id:	7BZdhyKy8pa0auAqaDB0geUkRO4oVEy4u-nQWRWtNcmJ3QQ0DSVmYw==
X-Amz-Cf-Pop:	DUB56-P3
X-Cache:	Hit from cloudfront

▸ Request Headers (19)

```
****************************************************************
*** Newer version of CDK is available [2.1007.0] ***
```

Demo



https://d2zp5eo3olaezx.cloudfront.net

```
Name
d2zp5eo3olaezx.cloudfront....
d2zp5eo3olaezx.cloudfront....
index-67e185b7.js
index-94c845ce.css
react-35ef61ed.svg
vite.svg
vite.svg
```

```
Last-Modified:                Mon, 07 Apr 2025
                              07:29:28 GMT
Server:                       AmazonS3
Vary:                         accept-encoding
Via:                          1.1
                              3d9f8f704e89e70cf394
                              d7dafb3f9fe4.cloudfron
                              t.net (CloudFront)
X-Amz-Cf-Id:                  vX6AgleApIT6C6ctcYUL
                              4axHbD_S3t_FtZghyWk
                              jDUmlnrkA8HklaQ==
X-Amz-Cf-Pop:                 DUB56-P3
X-Amz-Server-Side-            AES256
Encryption:
X-Cache:                      Miss from cloudfront
```

7 requests | 56.9 kB transferred | ▸ Request Headers (17)

```
*****************************************************
*** Newer version of CDK is available [2.1007.0] ***
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon 7 Apr 08:33
Stacks | CloudFormation | eu-
Mail - Diarmuid O'Connor - O
Channel dashboard - YouTube
Vite + React + TS
Dimensions: Responsive 400 x 490 100% No throttling
Elements Console Sources Network Performance
Preserve log Disable cache No throttling
Filter Invert More filters
All Fetch/XHR Doc CSS JS Font Img Media Manifest WS Wasm Other
100 ms 200 ms 300 ms 400 ms 500 ms 600 ms
Headers Preview Response Initiator
Count is: 0
OUTLINE
TIMELINE
FrontendOnly

CloudFront Cache

- A cache is a high-performance key-value store.
- The CF cache lives at each CloudFront Edge Location
- CloudFront identifies each object in the cache using the Cache Key
- We want to maximize the Cache Hit ratio, thus minimizing requests to the origin.
- We can invalidate parts of the cache using the CreateInvalidation API.

CloudFront – Cache keys

- Cache keys are unique identifiers for objects in the cache store.

- (Default) Key = Hostname + Resource portion of URL
- E,g, HTTP Get http://donainX.com/blog/article20.html?ref=123 → Key = donainX.com/blog/article20.html

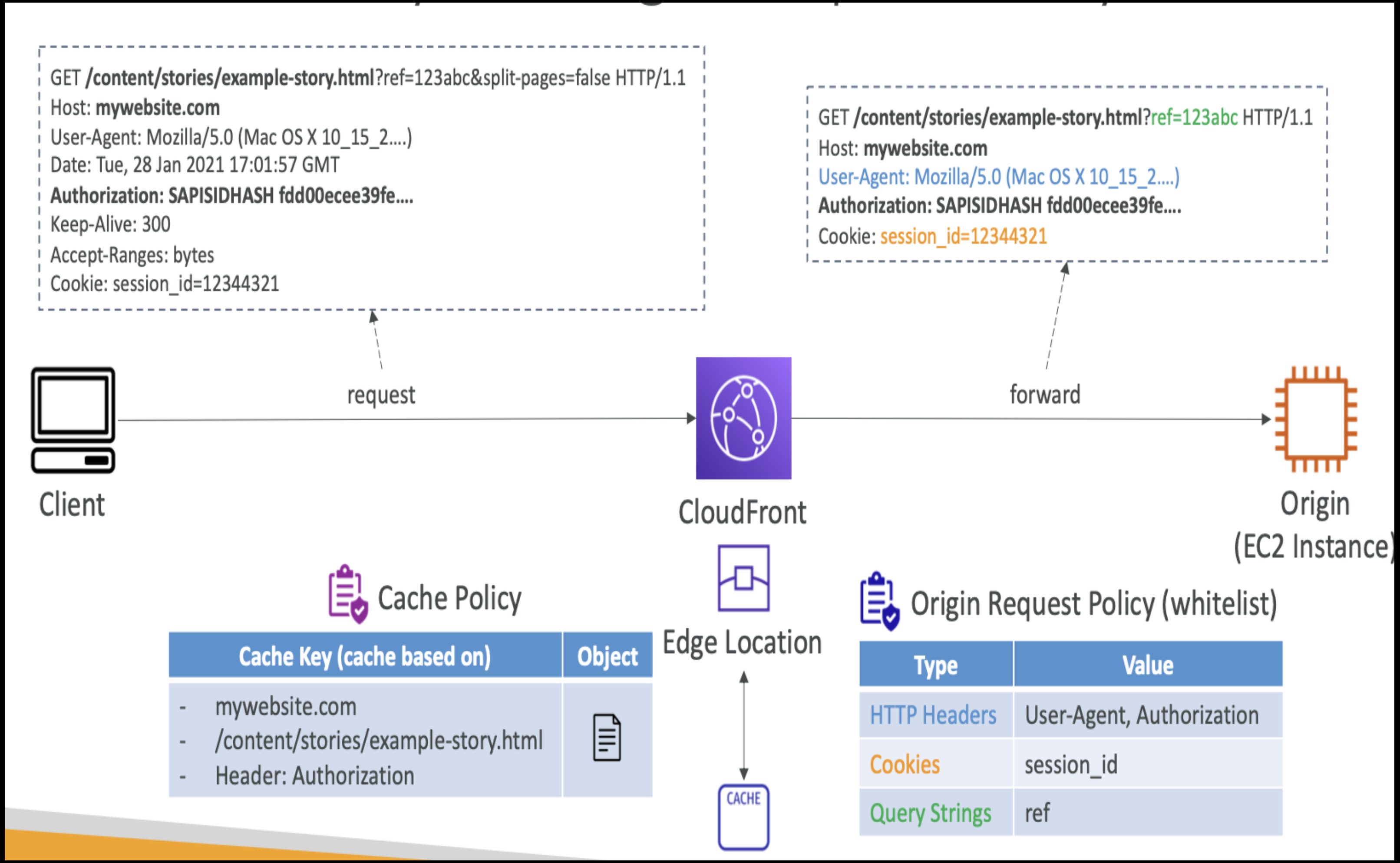
- If an application serves up content that varies based on user, device, language, location, etc, you can add other elements (i.e. HTTP headers, cookies, query strings) to the Cache Key using CloudFront Cache Policies.

CloudFront – Cache Policies

1. Cache policy.
   - Key based on:
     - HTTP Headers: None; Whitelist.
     - Cookies: None; Whitelist; Include All-Except All.
     - Query Strings: None; Whitelist; Include All-Except; All
   - Control the TTL (0 seconds to 1 year).

2. Origin Request policy.
   - Specify values that you want to include in origin requests without including them in the Cache Key.
     - HTTP headers: None; Whitelist; All.
     - Cookies: None; Whitelist; All.
     - Query Strings: None; Whitelist; All.

CloudFront – Cache Policies



```
GET /content/stories/example-story.html?ref=123abc&split-pages=false HTTP/1.1
Host: mywebsite.com
User-Agent: Mozilla/5.0 (Mac OS X 10_15_2....)
Date: Tue, 28 Jan 2021 17:01:57 GMT
Authorization: SAPISIDHASH fdd00ecee39fe....
Keep-Alive: 300
Accept-Ranges: bytes
Cookie: session_id=12344321
```

```
GET /content/stories/example-story.html?ref=123abc HTTP/1.1
Host: mywebsite.com
User-Agent: Mozilla/5.0 (Mac OS X 10_15_2....)
Authorization: SAPISIDHASH fdd00ecee39fe....
Cookie: session_id=12344321
```

Cache Policy

Cache Key (cache based on)	Object
- mywebsite.com - /content/stories/example-story.html - Header: Authorization	

Origin Request Policy (whitelist)

Type	Value
HTTP Headers	User-Agent, Authorization
Cookies	session_id
Query Strings	ref

CloudFront – Cache Invalidation.

- Problem: After updating the back-end origin, CloudFront doesn’t know about it and will only get the updated content after the TTL has expired.
- Solution: Force an entire or partial cache refresh (thus bypassing the TTL) by performing a CloudFront Invalidation.
- You can invalidate all files (*) or a special path (/images/*).

Full-stack web App deployment.

- Objective: Use the CDK framework to provision the cloud infrastructure for a Full-stack web app.
  - How does the frontend resolve the backend API(s) URL?
    - Manually – Hardcoded.
    - Automatically - Dynamically at deployment time.

Custom Domains

- Objective Associate a custom domain name with a Cloudfront-enamled web app.
- Steps:

1. Buy a domain name - see https://porkbun.com/ - Watch - https://www.youtube.com/watch?v=kl3a76CBwX4
2. In the AWS Route53 console, create a Hosted Zone for your domain name. (Manual)
3. Copy the hosted zone's list of name servers to your registered domain name on its Domain Registry (e.g. porkbun). (Manual)
4. To support HTTPS traffic, create a Certificate for your domain/subdomain using the AWS Certificate Manager service in the us-east-1 region. (CDK)
5. Attach the certificate to your Cloudfront distribution for the particular domain/subdomain. (CDK)
6. Add an A record to your hosted zone to redirect requests to your app's custom URL to the CF distribution. (CDK)