

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.

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

```
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,
  });
}
```



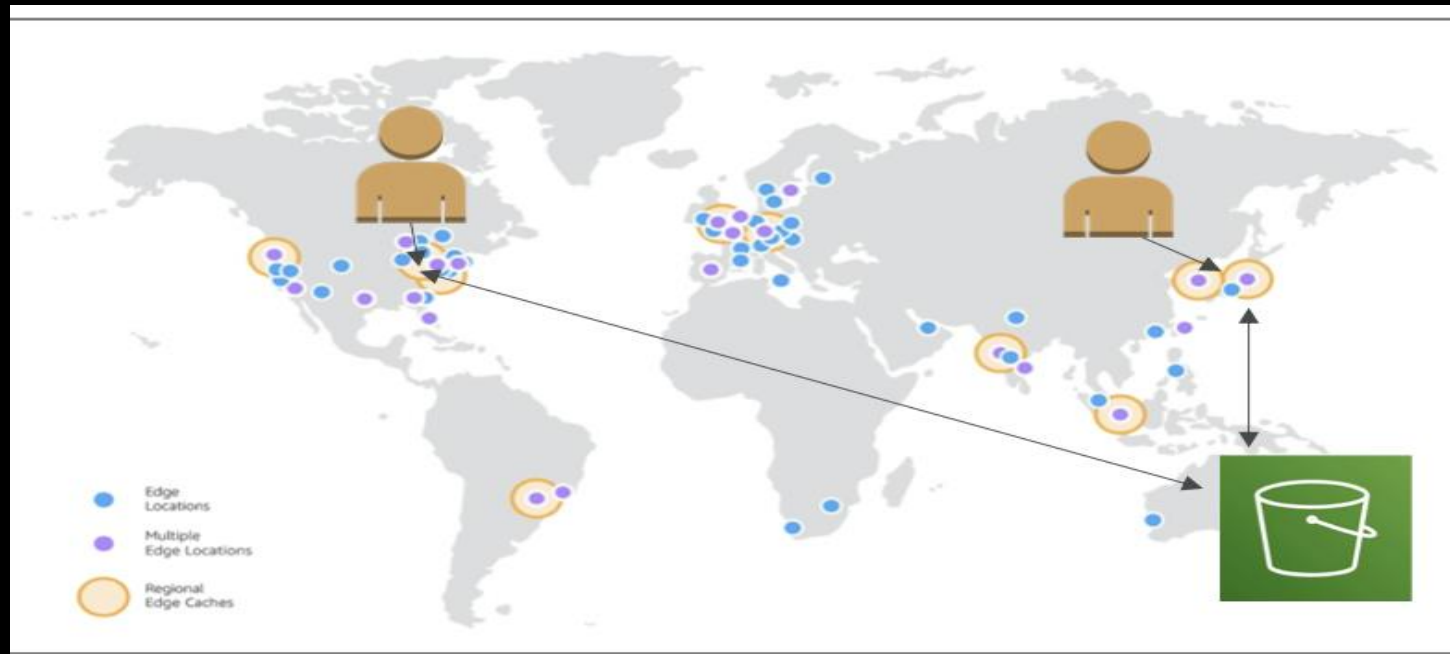
AWS CloudFront (CDN)

CloudFront / CDN

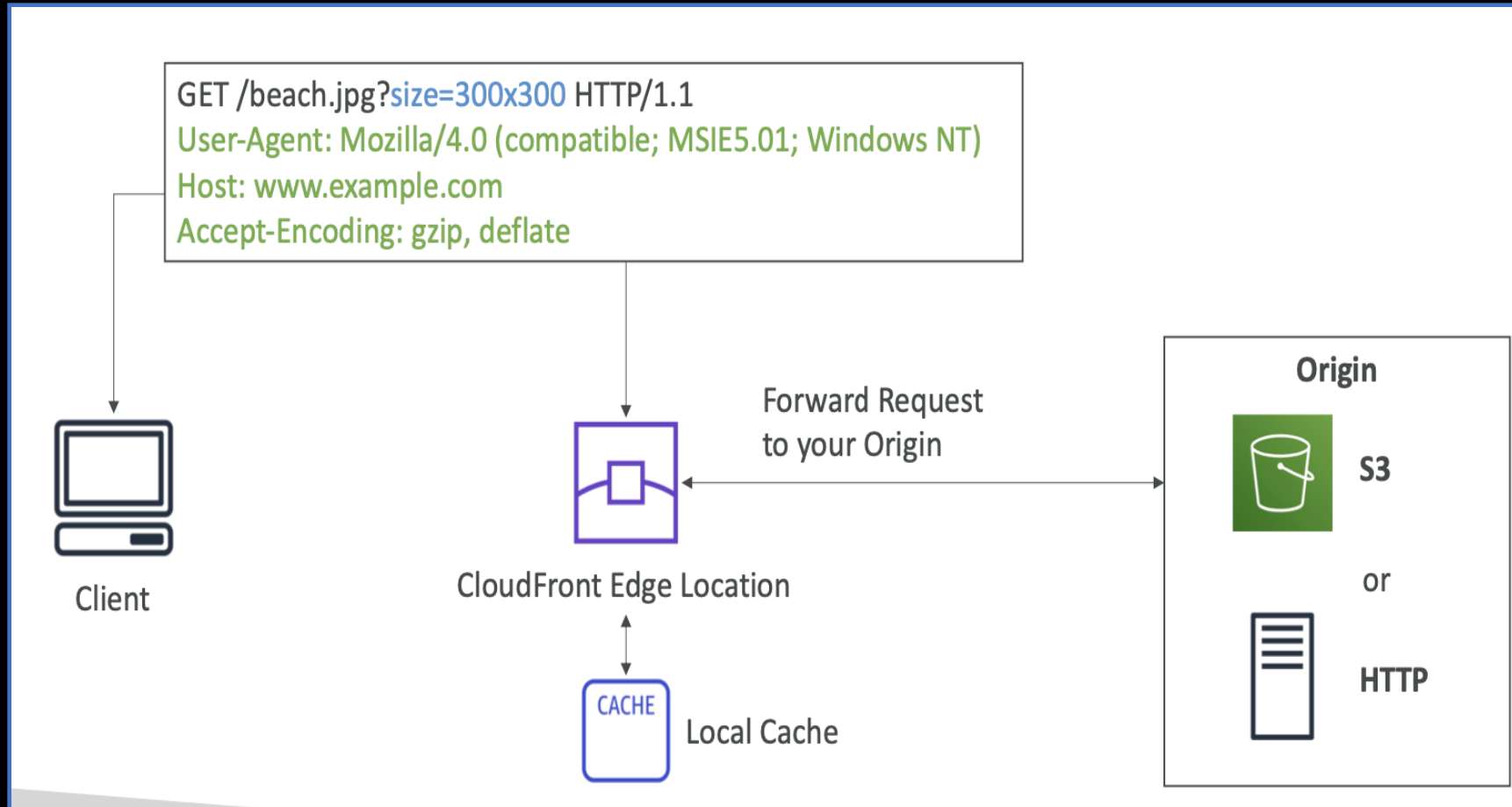
- A CDN (Content Delivery Network) is a network of geographically distributed servers that cache copies of content close to the end users, thus lowering latency when they download or stream content.
 - Servers = Edge locations or Points of Presence (POPs).
 - Content = Web assets, Documents, Videos,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

CloudFront - 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.



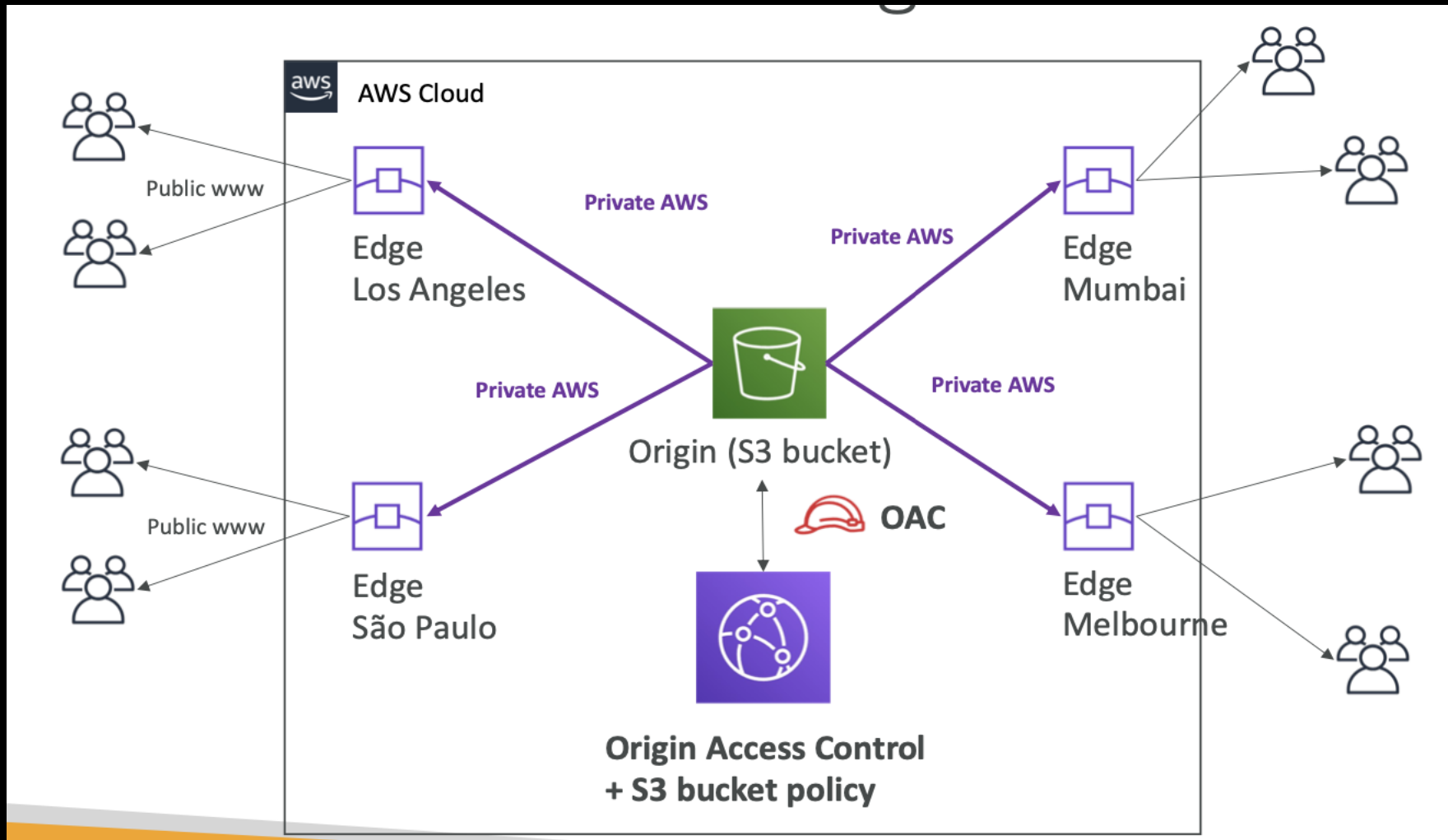
CloudFront resources



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

Chrome | File | Edit | View | History | Bookmarks | Profiles | Tab | Window | Help | Mon 7 Apr 08:35

Stacks | CloudFormation | eu- | Mail - Diarmuid O'Connor - | Channel dashboard - YouTube | Vite + React + TS

https://d2zp5eo3olaezx.cloudfront.net

Dimensions: Responsive | 400 x 490 | 100% | No throttling

Count is: 0

Network | Elements | Console | Sources | Performance

Filter | Invert | More filters

All | Fetch/XHR | Doc | CSS | JS | Font | Img | Media | Manifest | WS | Wasm | Other

50 ms | 100 ms | 150 ms | 200 ms | 250 ms | 300 ms

Name	Headers	Preview	Response	Initiator
d2zp5eo3olaezx.cloudfront....			07:34:42 GMT	
index-67e185b7.js	Etag:		"4c046a0d76d0ae72f3f08500ddd1db62"	
index-94c845ce.css				
react-35ef61ed.svg	Server:		AmazonS3	
vite.svg	Vary:		accept-encoding	
vite.svg	Via:		1.1	
			3d9f8f704e89e70cf394d7dafb3f9fe4.cloudfront.net (CloudFront)	
	X-Amz-Cf-Id:		7BZdhyKy8pa0auAqaDB0geUkRO4oVEy4u-nQWRWtNcmJ3QQ0DS	
	X-Amz-Cf-Pop:		DUB56-P3	
	X-Cache:		Hit from cloudfront	

6 requests | 1.6 kB transferred | Request Headers (19)

OUTLINE | TIMELINE

FrontendOnly | 0 | 0 | 0

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

Demo

Chrome | File | Edit | View | History | Bookmarks | Profiles | Tab | Window | Help | Mon 7 Apr 08:33

Stacks | CloudFormation | eu- | Mail - Diarmuid O'Connor - | Channel dashboard - YouTube | Vite + React + TS

https://d2zp5eo3olaezx.cloudfront.net

Dimensions: Responsive 400 x 490 100% No throttling

Count is: 0

Network

Filter: 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

Name	Headers	Preview	Response	Initiator
d2zp5eo3olaezx.cloudfront...	Last-Modified:		Mon, 07 Apr 2025	
d2zp5eo3olaezx.cloudfront...			07:29:28 GMT	
index-67e185b7.js	Server:		AmazonS3	
index-94c845ce.css	Vary:		accept-encoding	
react-35ef61ed.svg	Via:		1.1	
vite.svg			3d9f8f704e89e70cf394	
vite.svg			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)

OUTLINE

TIMELINE

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

FrontendOnly 0 0 0 0

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

- 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 key is a unique identifier for every object 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 keys

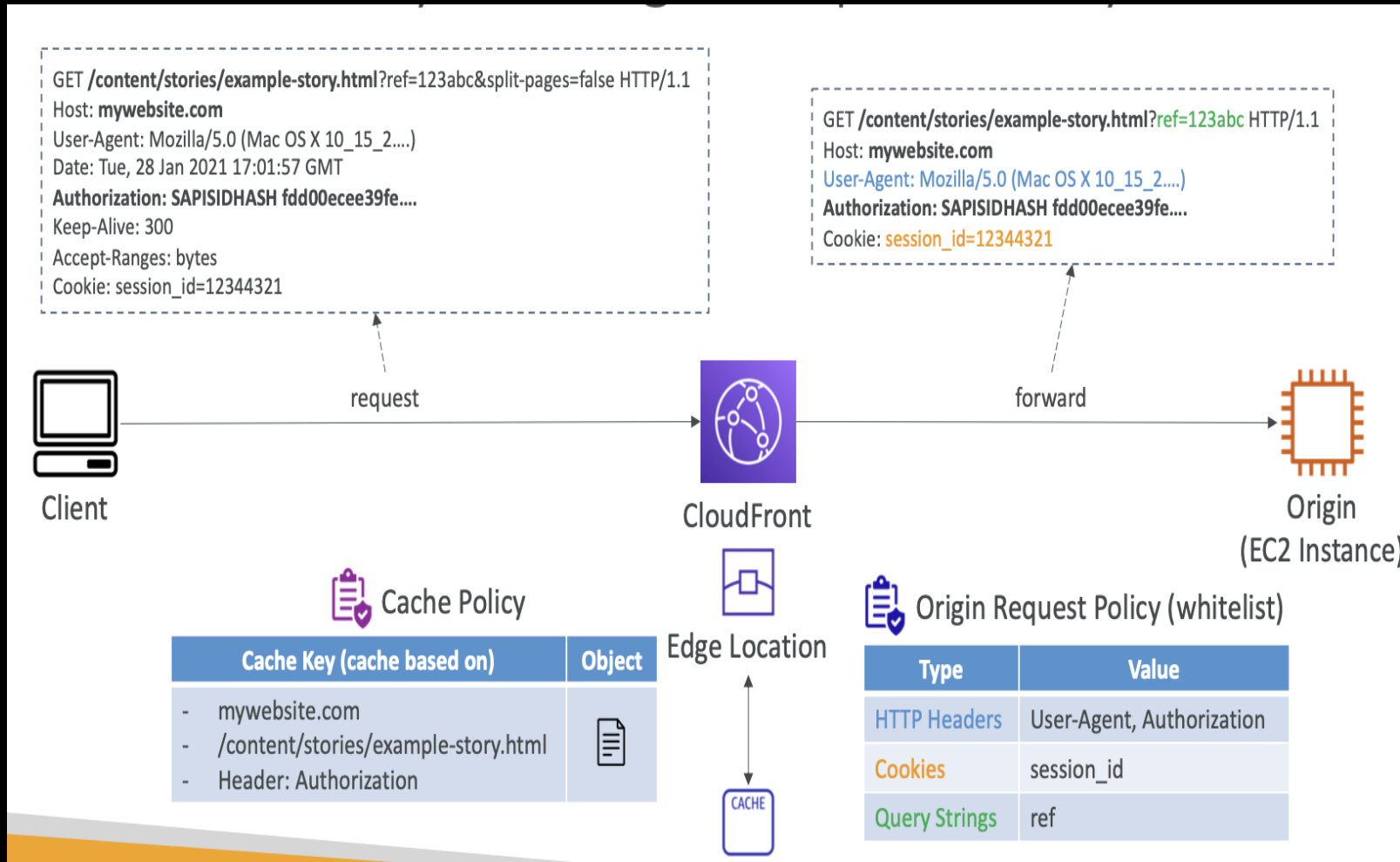
1. Cache policy.

- 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 keys



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-enabled 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)