salesqueze

Global Speed on Low-Traffic Routes with CloudFront

… or how we made
CF, Lambda@Edge, CF functions, CF KeyValue Store, S3, MRAP
work together!

November 26, 2025

Tilen Tomakić
www.linkedin.com/in/tomakic

Context

SaleSqueze is Multi-tenant SaaS

- Producing high dynamic content in a form of web configurators.
- Traffic can be uneven when you look at a single tenant.
- Users around the world.

What We Needed

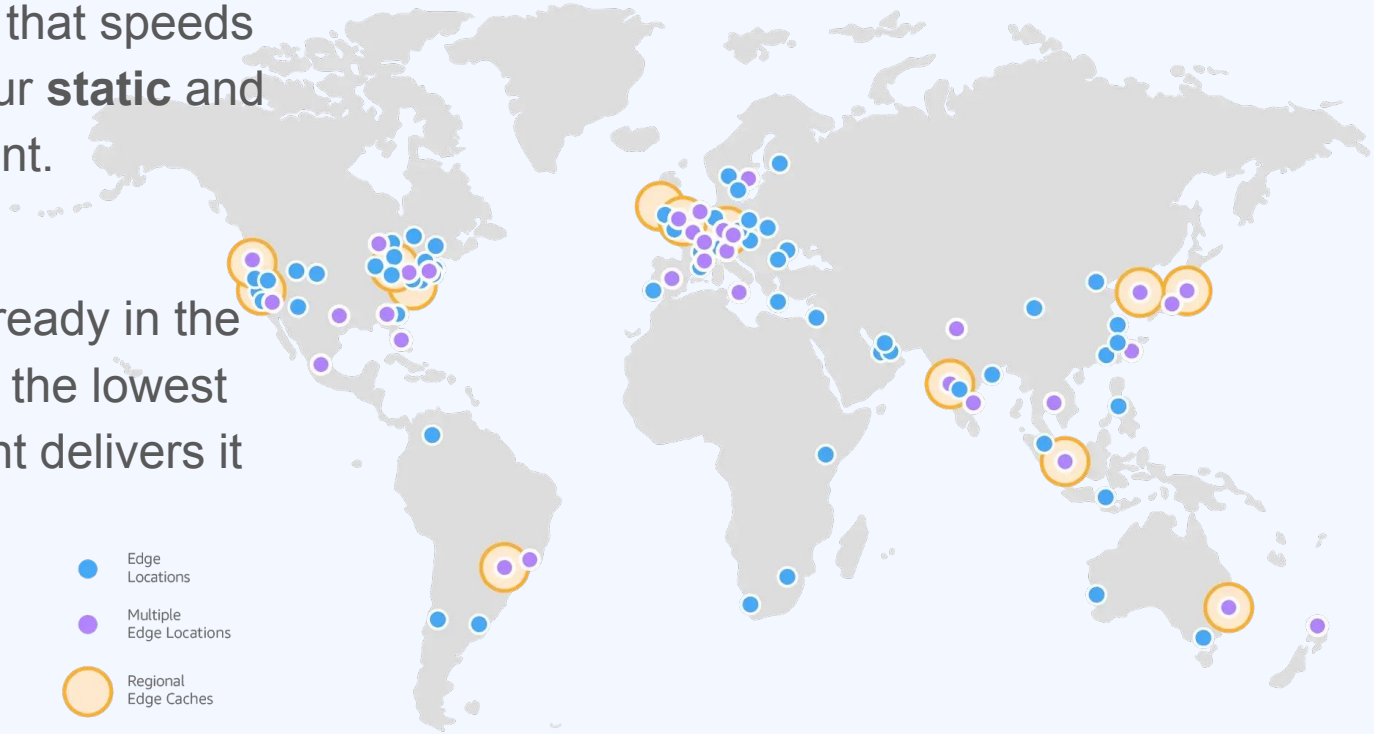
Goals for picking a CDN solution

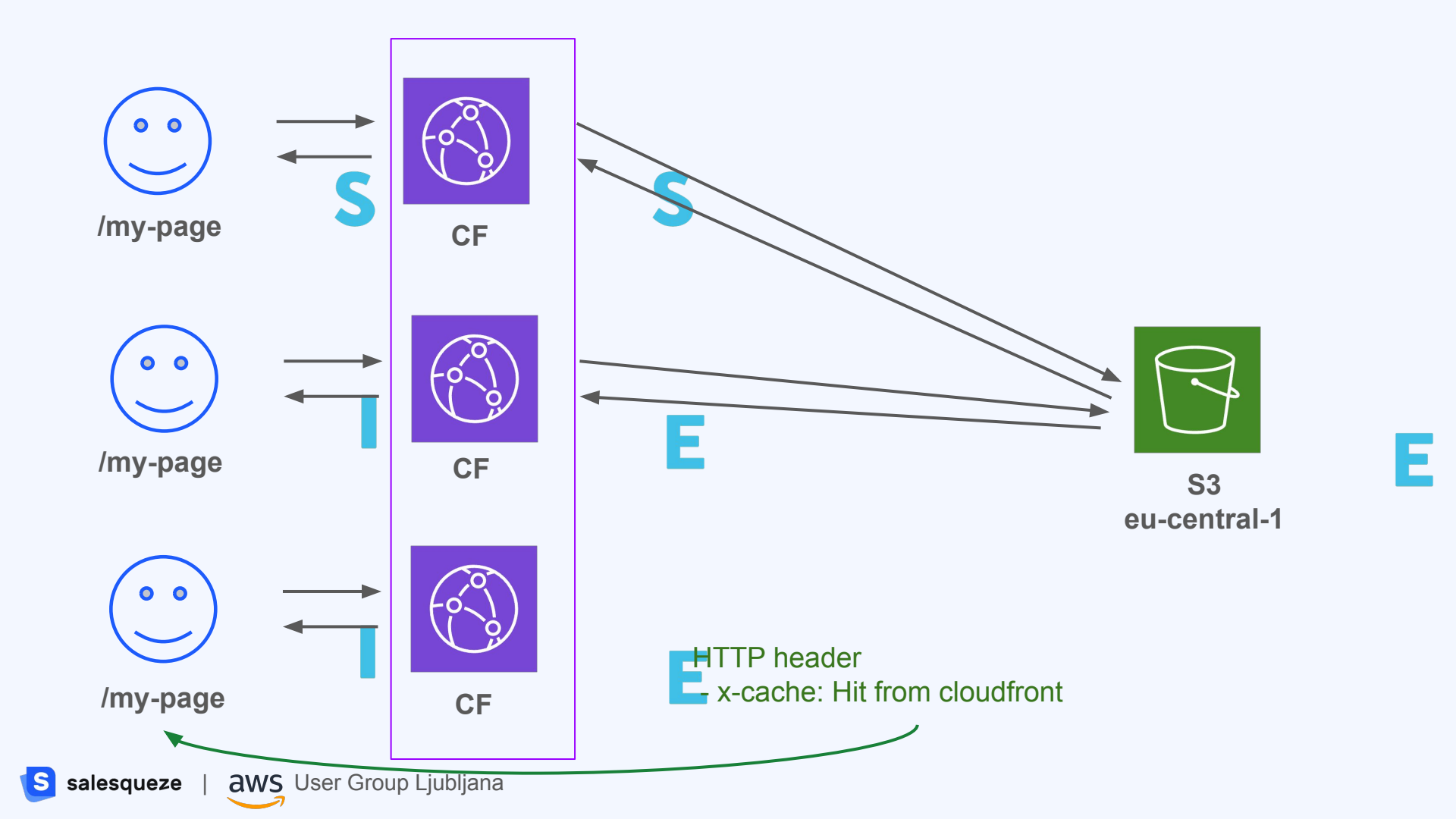
- Fast global response
- No dependency on warm cache
- Dynamic URL rewrites and redirects
- No mass invalidations
- Ideally, we stay inside the AWS and make it part of our AWS CDK

AWS CloudFront to the rescue?

CF is a web service that speeds up distribution of your **static** and **dynamic** web content.

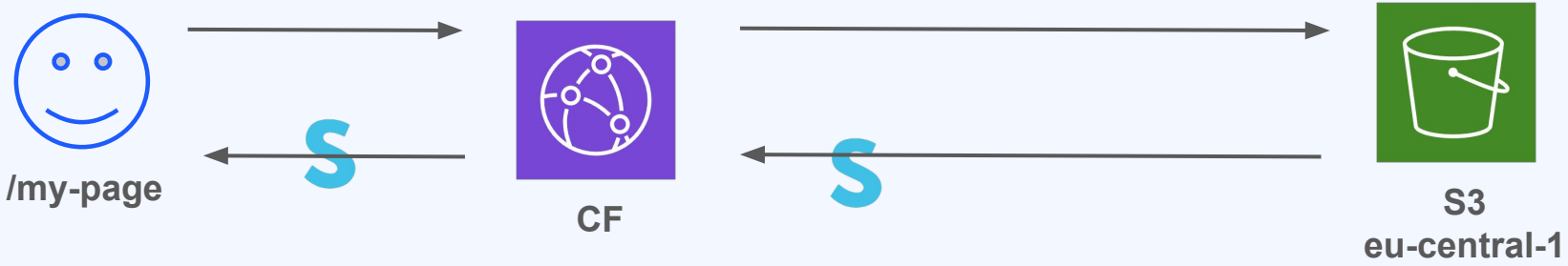
- If the content is already in the edge location with the lowest latency, CloudFront delivers it immediately.




/my-page
/my-page
/my-page
CF
CF
CF
S
S
I
I
E
E
E
S3
eu-central-1
HTTP header
- x-cache: Hit from cloudfront

How to do URL rewrites and redirects?

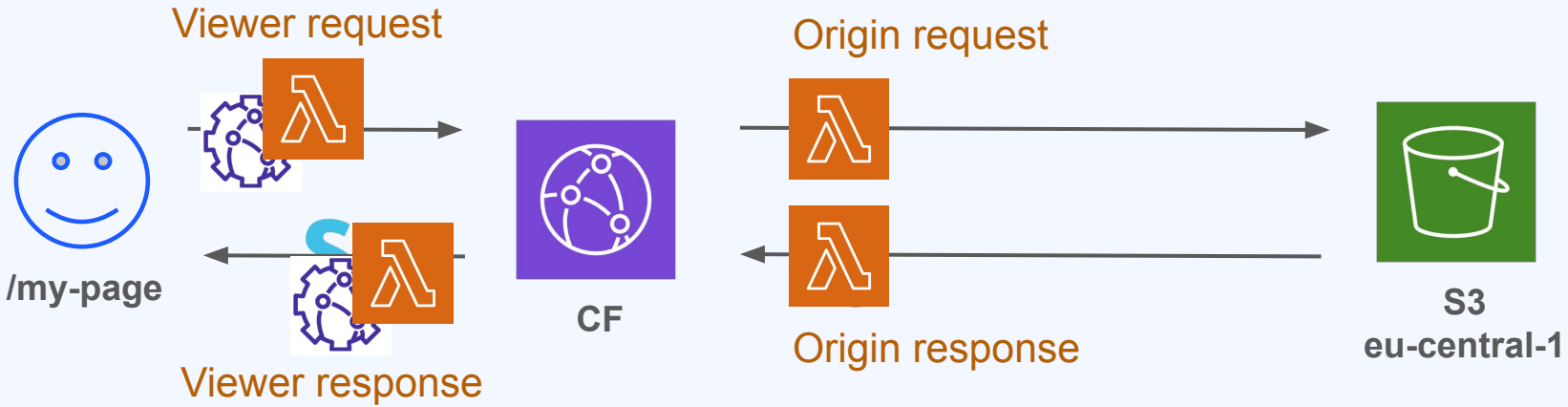
Function associations to the rescue!



S3 contents could be:
/public/my-page.html

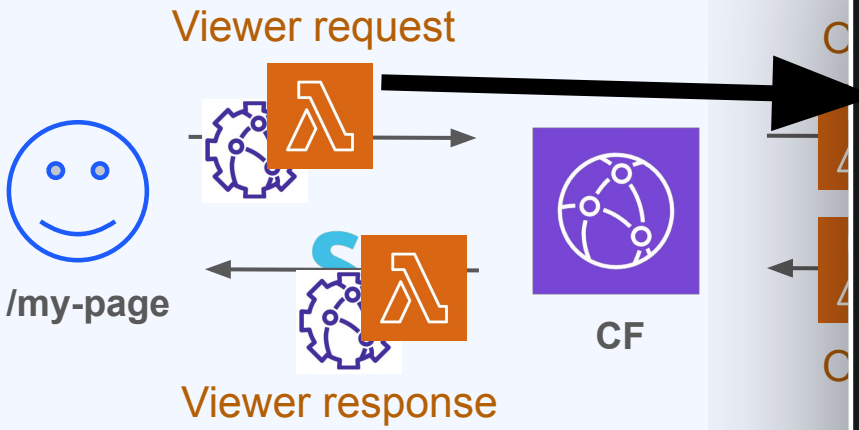
How to do URL rewrites and redirects?

Function associations to the rescue!



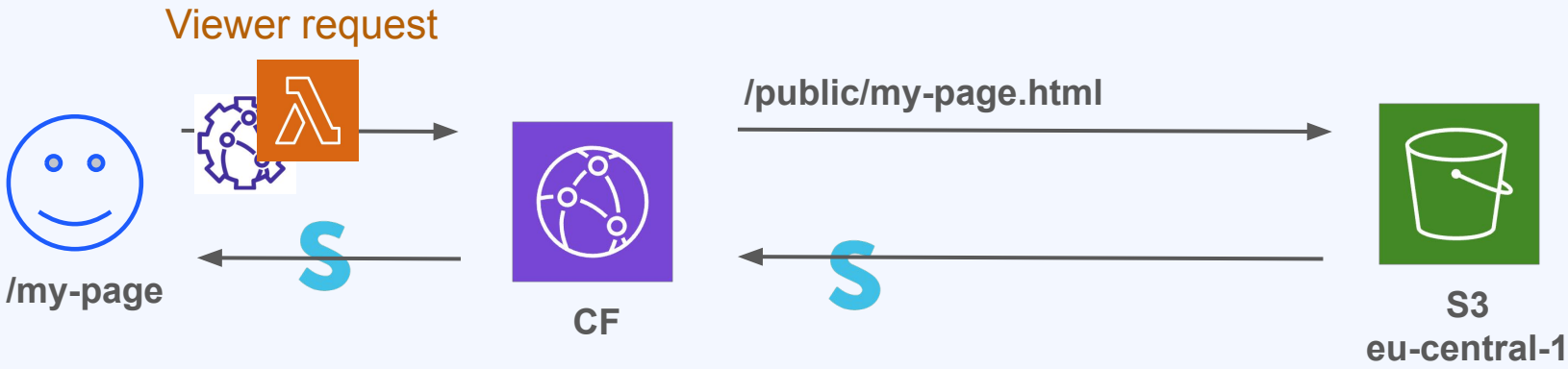
S3 contents could be:
/public/my-page.html

How to do URL rewrites and redirects?




```
async function handler(event) {
  var request = event.request;
  var uri = request.uri;

  if (!uri.includes('.')) {
    request.uri = '/public'
      + request.uri
      + '/index.html';
  }

  return request;
}
```

How to do URL rewrites and redirects?



S3 contents could be:
/public/my-page.html

But what about **Dynamic** URL rewrites and redirects?

For example, we want:

a-tenant.salesqueze.com/configurator
to point on S3 to:
/cache/2025/11/26/b882c6.html

and

b-tenant.salesqueze.com/configurator
to point on S3 to:
/cache/2025/11/20/af7d35.html

But these pointers can change at any time!

What are the options?

- Invalidate CF cache on every change
- Re-deploy lambda on every change
- Use some sort of DB and make lambda lookup current pointer ✅

But what about **Dynamic** URL rewrites and redirects?

Table

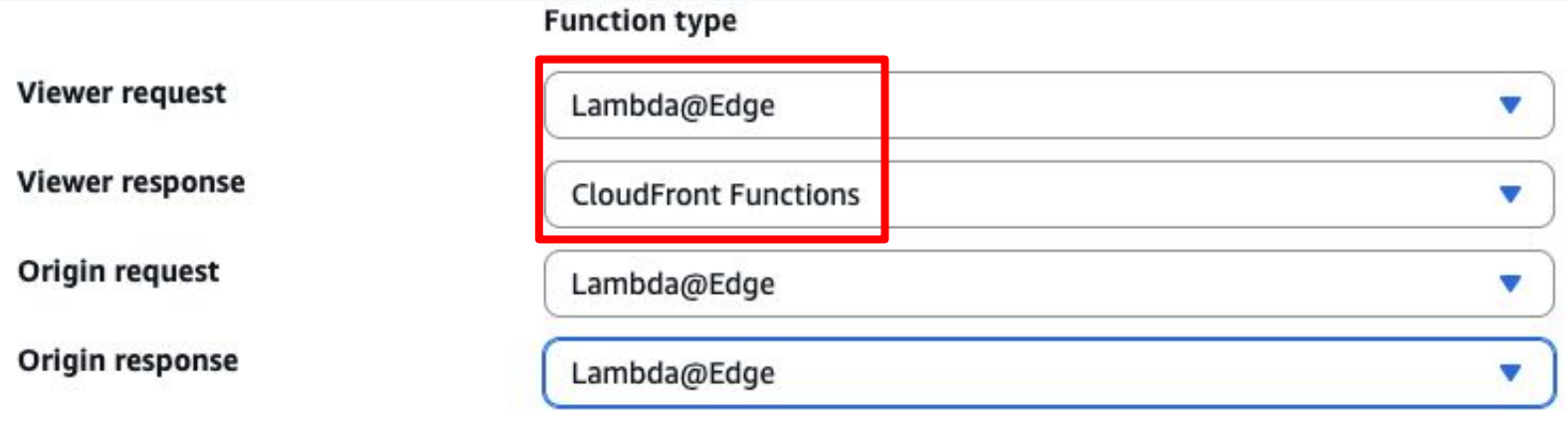
host	S3 location
a-tenant	**/cache/2025/11/26/b882c6.html**
b-tenant	**/cache/2025/11/20/af7d35.html**
…	**…**

But what DB to pick? We need:

- Fast response times
- Can survive spikes
- No connection overhead

Answer:
DynamoDB

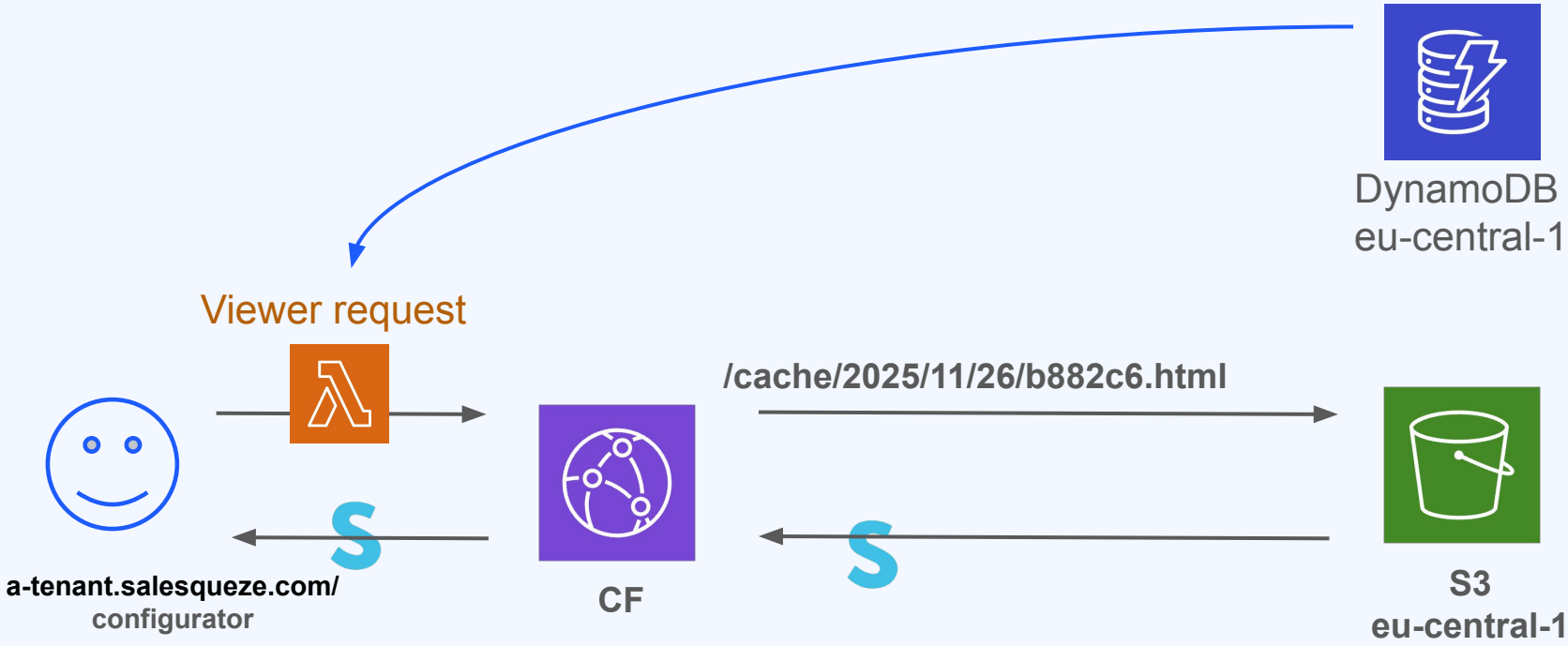
Function association types



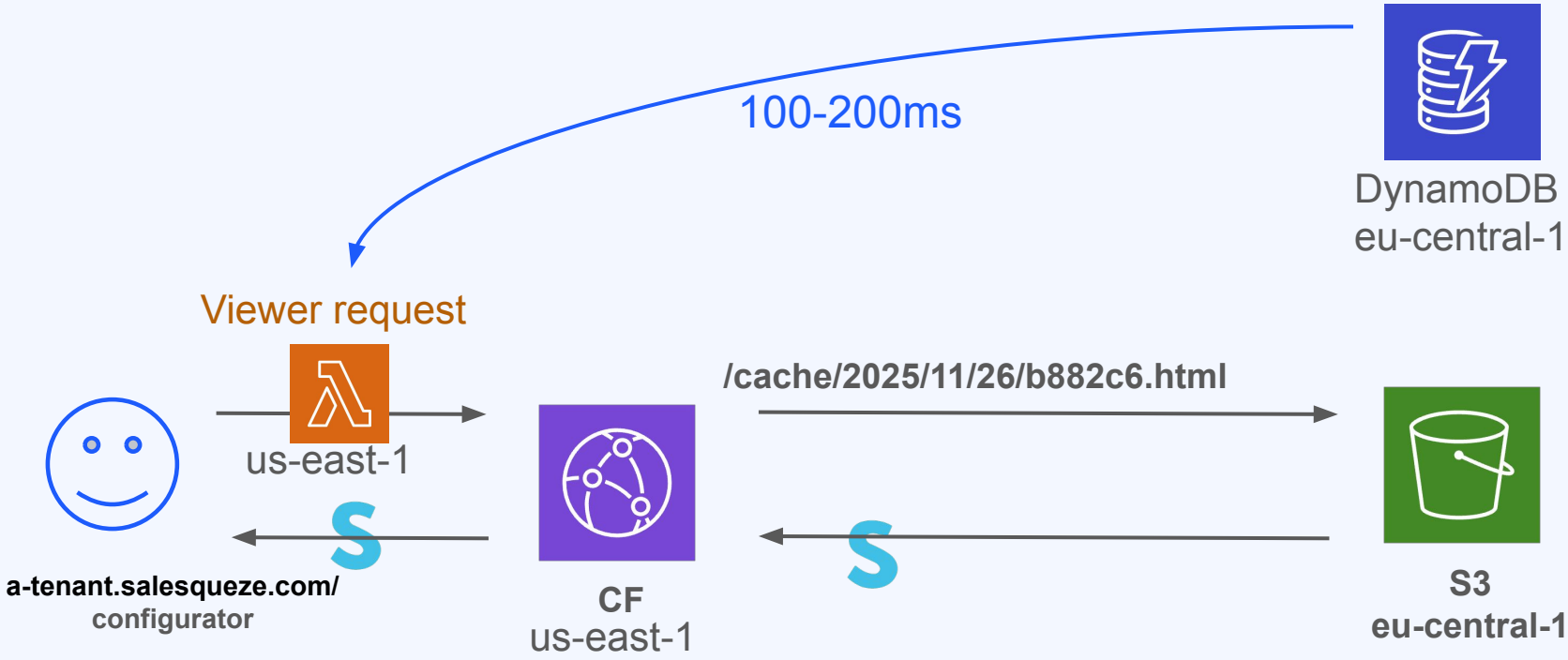
What's the difference?

	CloudFront Functions	**Lambda@Edge**
Programming languages	JavaScript	Node.js and Python
Event sources	• Viewer request • Viewer response	• Viewer request • Viewer response • Origin request • Origin response
Scale	Up to **millions** of requests per second	Up to 10,000 requests per second per Region
Maximum size of code	**10 KB**	50 MB
Network access	**No**	Yes
Function duration	**Submillisecond**	Up to 30 seconds

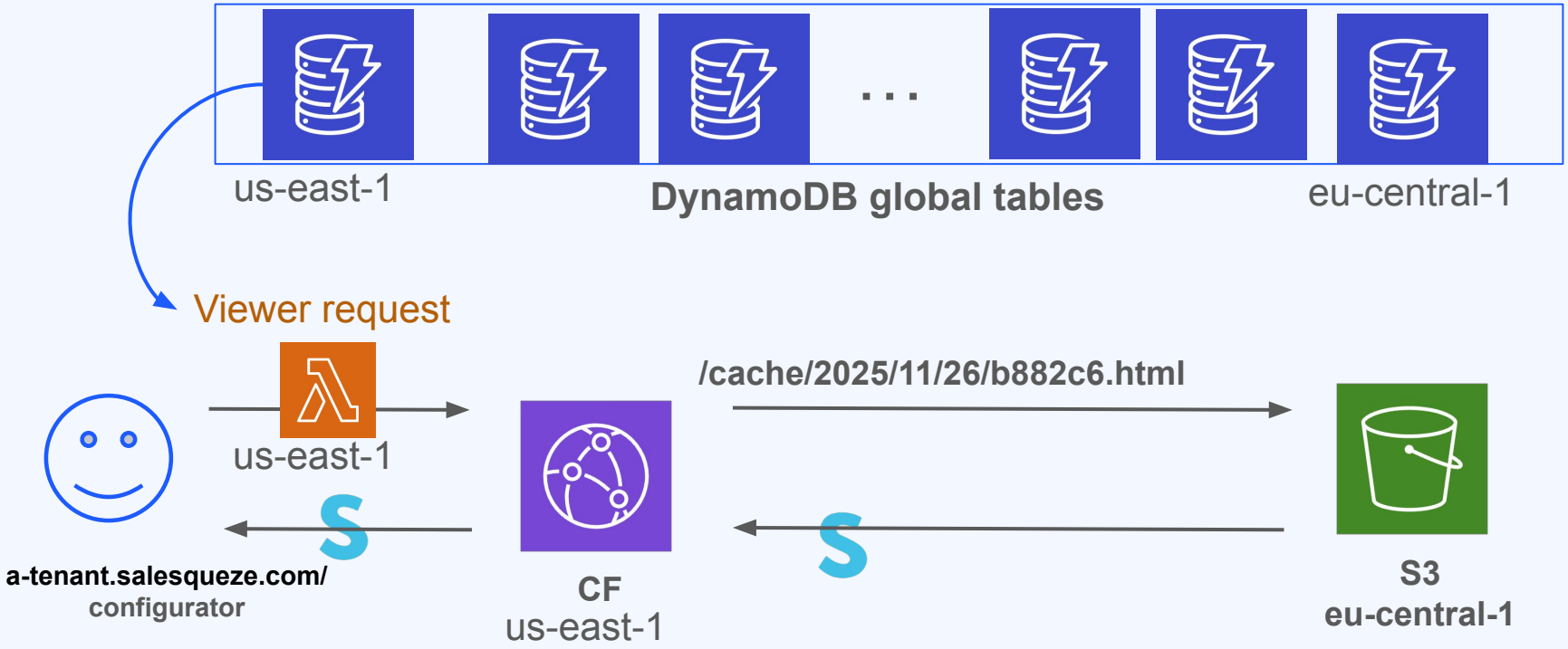
How to do URL rewrites and redirects?



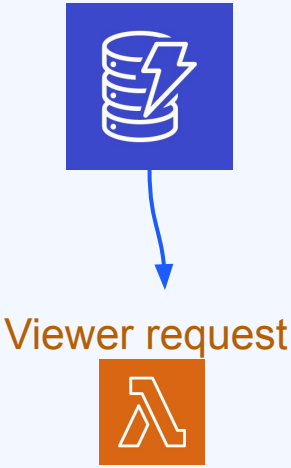
How to do URL rewrites and redirects?



How to do URL rewrites and redirects?

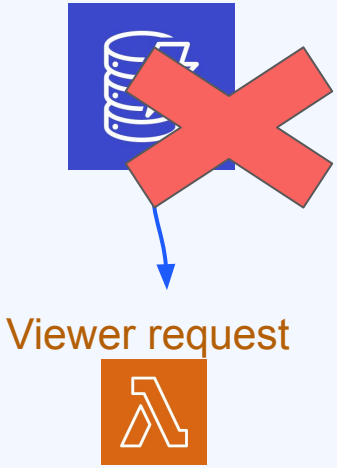


Could it be better?



- ✓ Dynamic routing
- x Fetching routing data from DB still adds 30-50ms delay

Could it be better?



- ✓ Dynamic routing
- x Fetching routing data from DB still adds 30-50ms delay

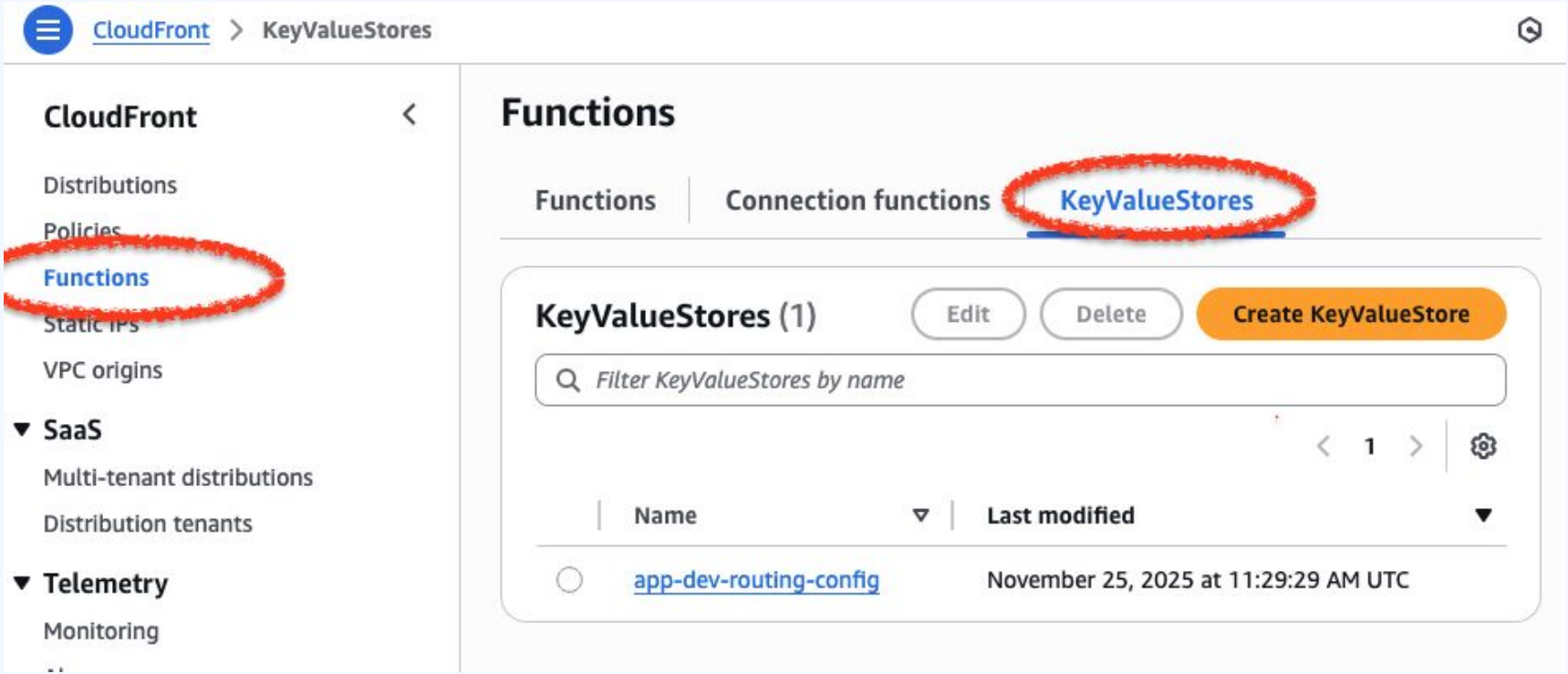
In November 2024 AWS introduced:

CloudFront KeyValueStore

A low-latency datastore for CloudFront Functions

- Fast, low-latency lookup at the edge
- Great for tenant-level routing rules
- Lets you avoid running Lambda@Edge for everything
- Useful for dynamic path resolution
- **Can only be used in CloudFront functions**

Where can you find KeyValueStores?



CloudFront KeyValue Store

Pricing

- Read pricing: $0.03 per 1 million reads from within CloudFront Functions.
- **Non-read API actions: $1 per 1,000 API requests.**

CloudFront Functions

- **JavaScript (ECMAScript 5.1 compliant)**
- Only lib available is AWS Helper methods

```
import cf from 'cloudfront';
```

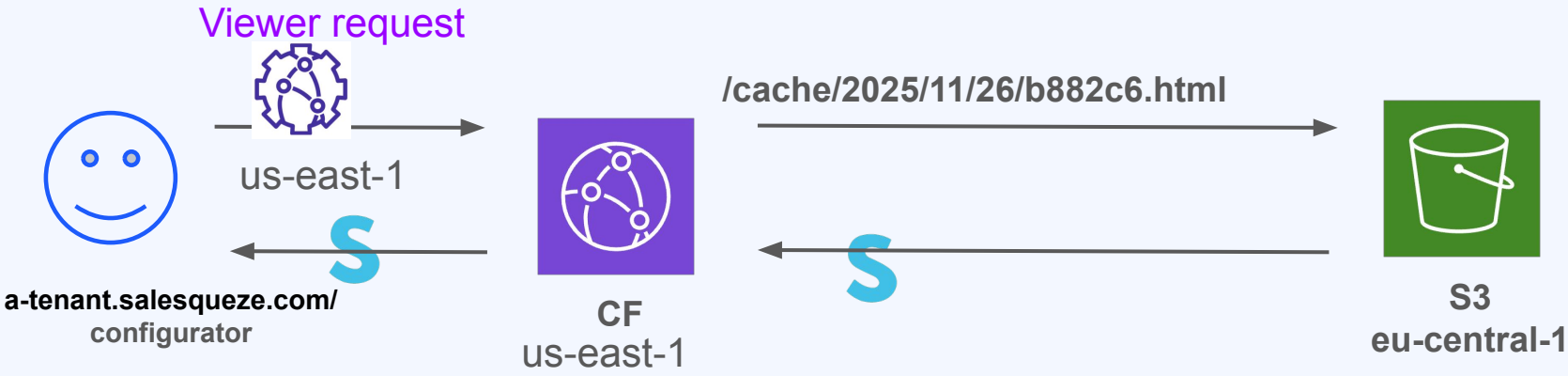
Maximum size of code	**10 KB**
Network access	**No**
Function duration	**Submillisecond**

Reading from CloudFront KeyValueStore

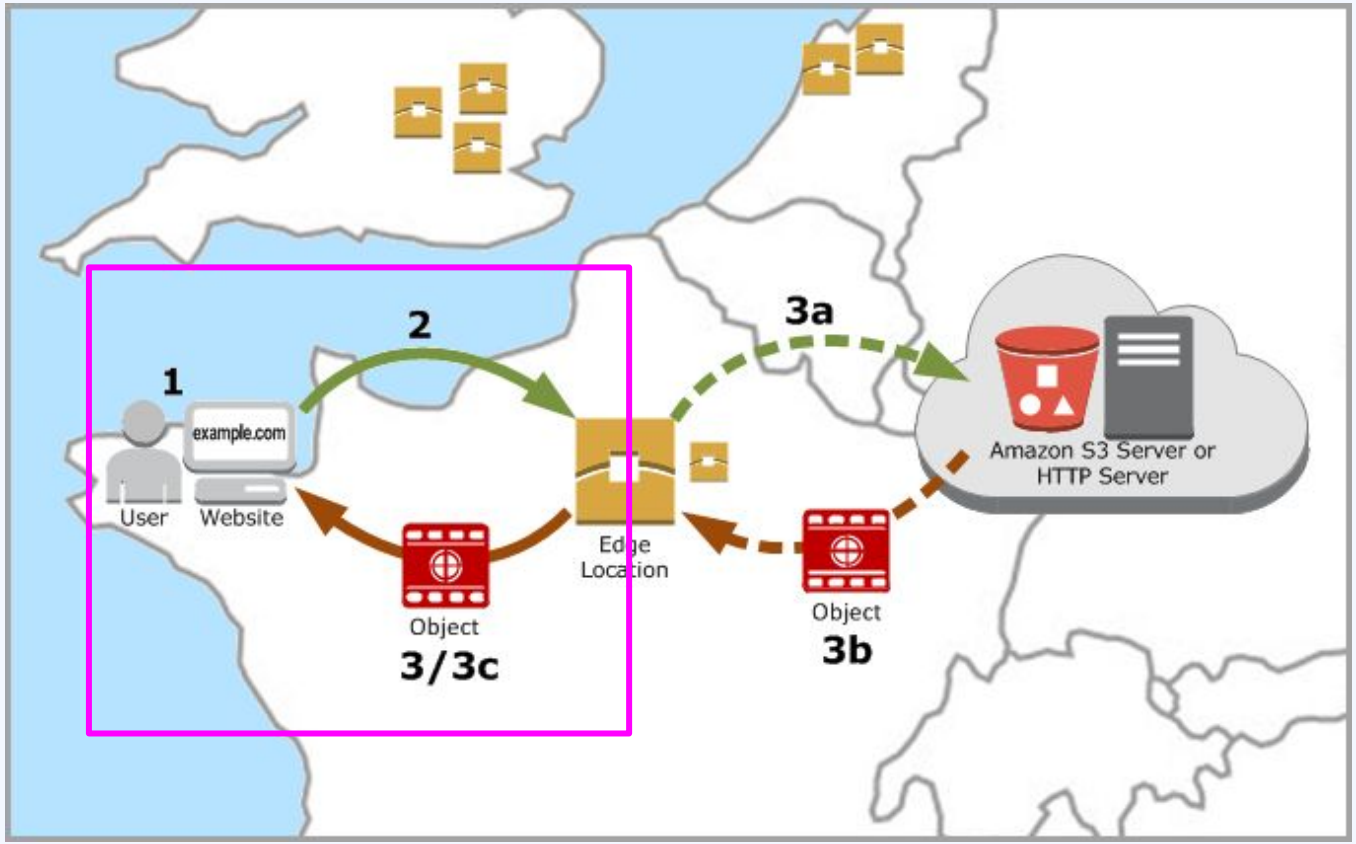
```
import cf from 'cloudfront';
…
const kvsHandle = cf.kvs();
const value = await kvsHandle.get("a-tenant", { format: "string"});
```

```
var values = await Promise.all([
  kvs.get('key1'),
  kvs.get('key2')
]);
```

How URL rewrites and redirects look in CF functions

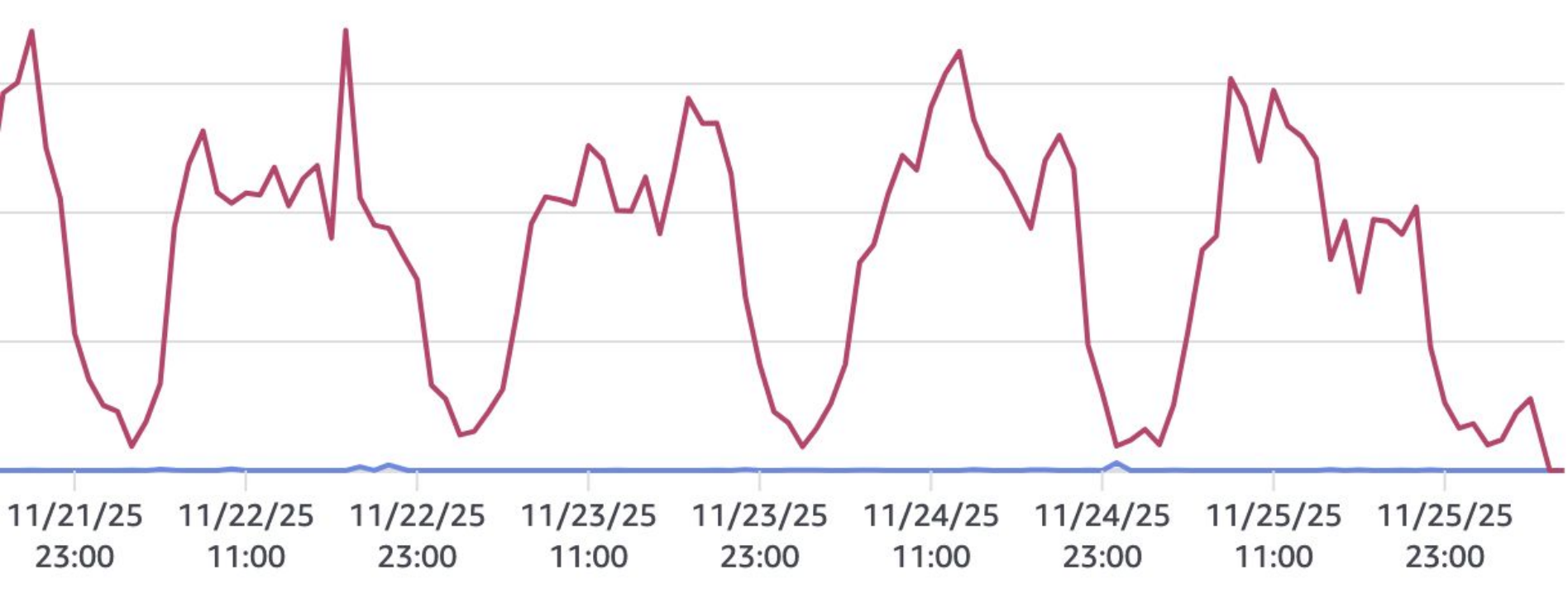


We optimized left part, what about right?



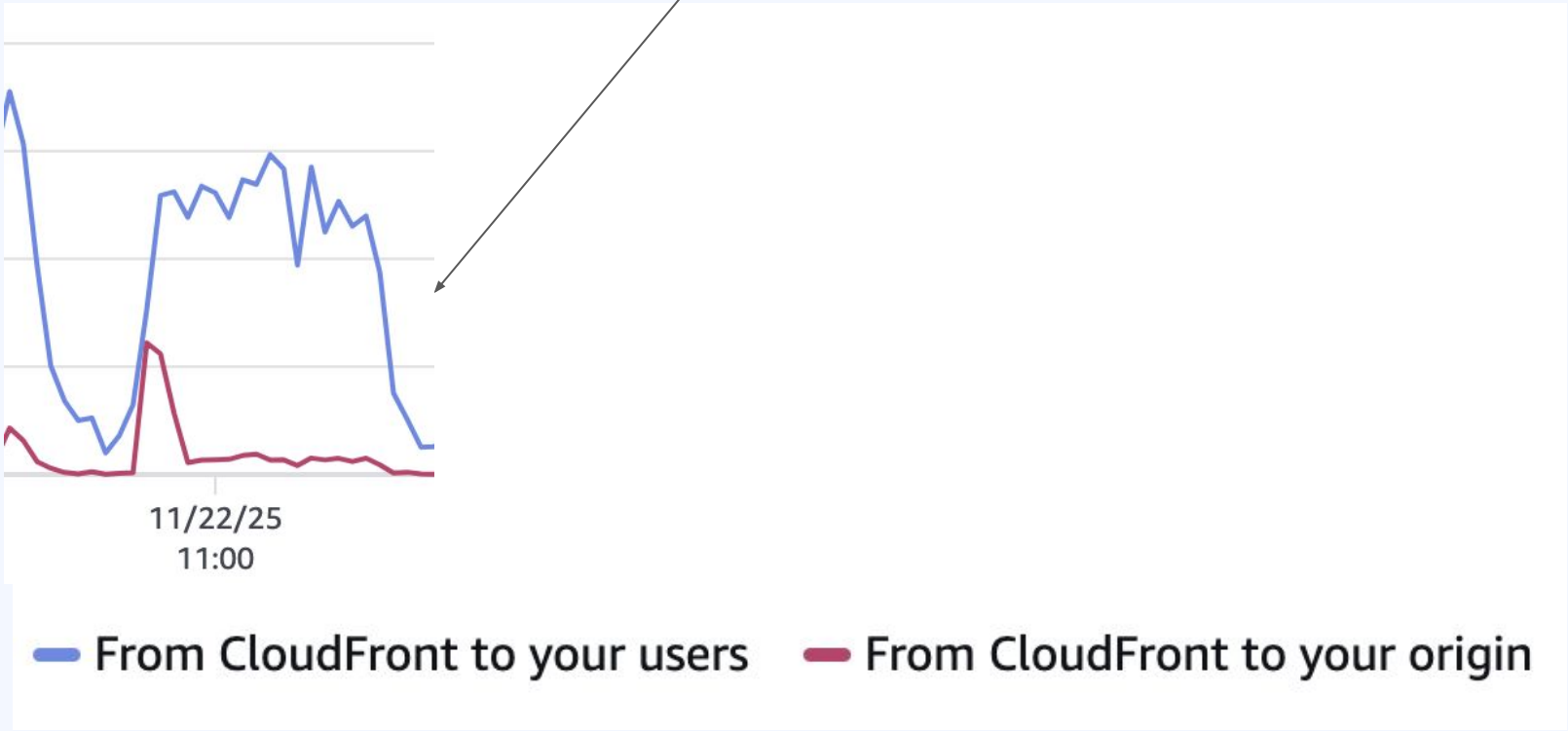
Number of requests

Statistics for one tenant



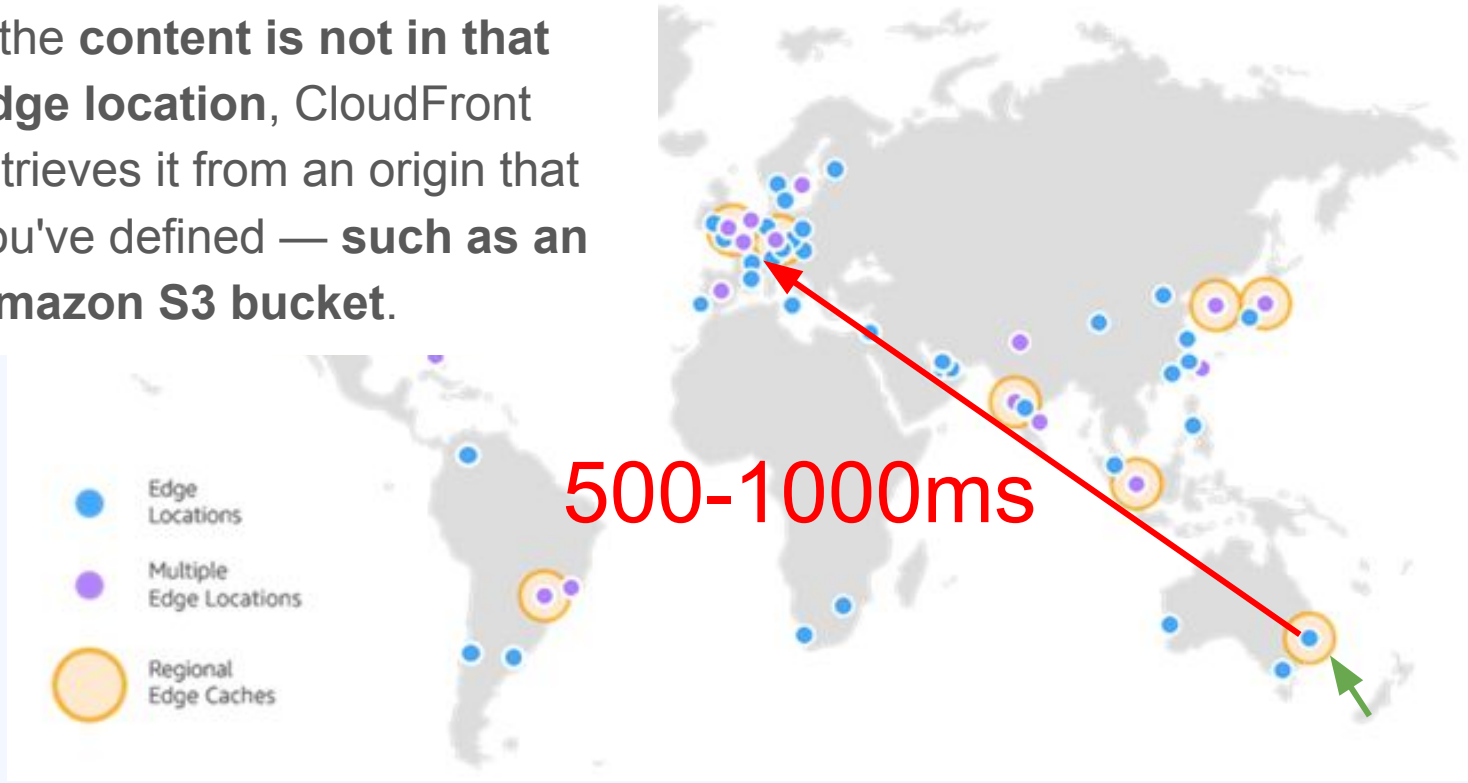
The issue

Cache miss on everything at morning hours



Why is that a problem in our edge case?

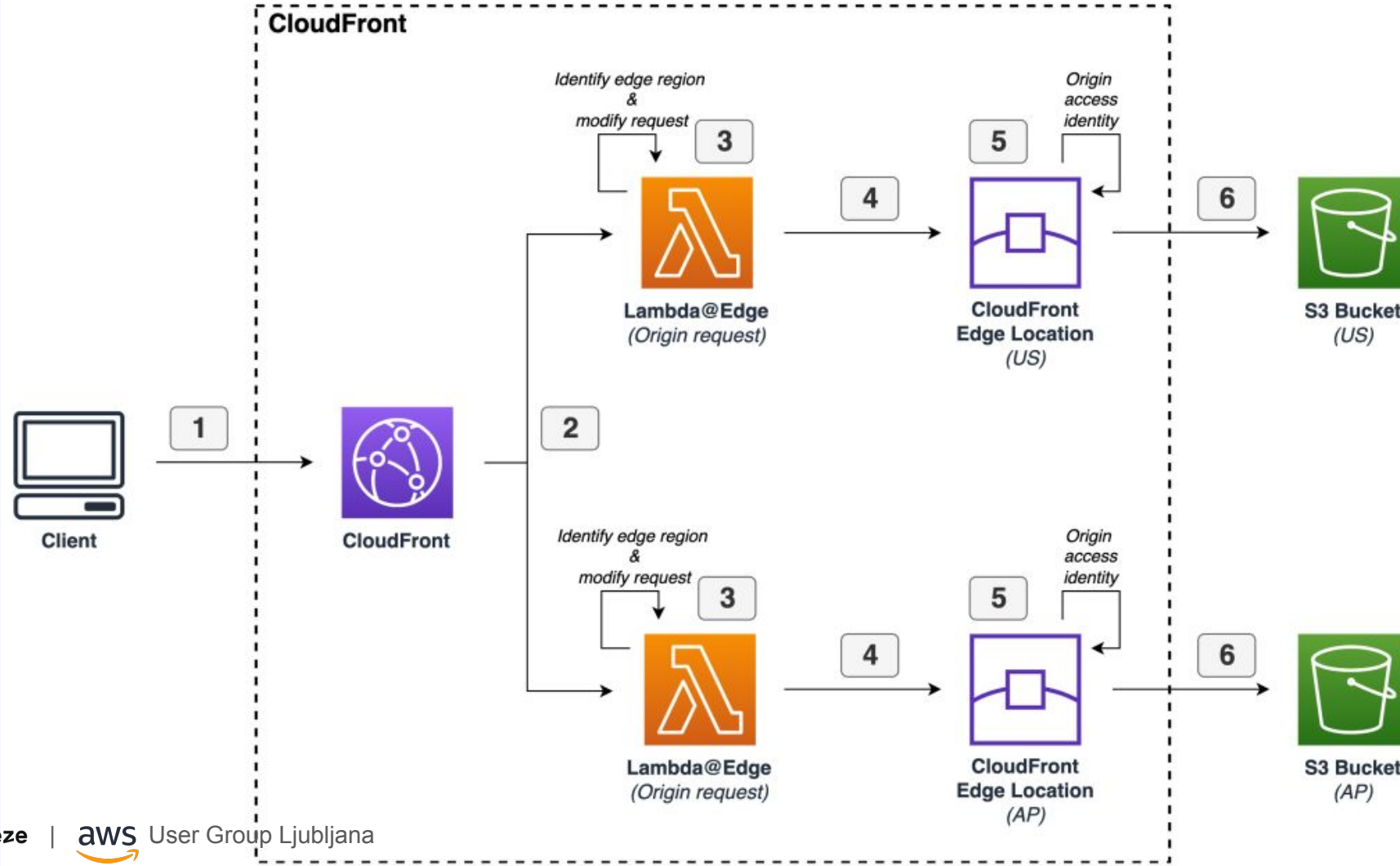
- If the **content is not in that edge location**, CloudFront retrieves it from an origin that you've defined — **such as an Amazon S3 bucket**.



How to achieve fast global response on low traffic files?

What you will find on the internet is post:
Using Amazon CloudFront and Amazon S3 to build multi-Region active-active geo proximity applications

.... but


CloudFront
Identify edge region & modify request
3
Origin access identity
5
4
6
Lambda@Edge
(Origin request)
CloudFront Edge Location
(US)
S3 Bucket
(US)
1
2
Client
CloudFront
Identify edge region & modify request
3
Origin access identity
5
4
6
Lambda@Edge
(Origin request)
CloudFront Edge Location
(AP)
S3 Bucket
(AP)

Regions Mapping

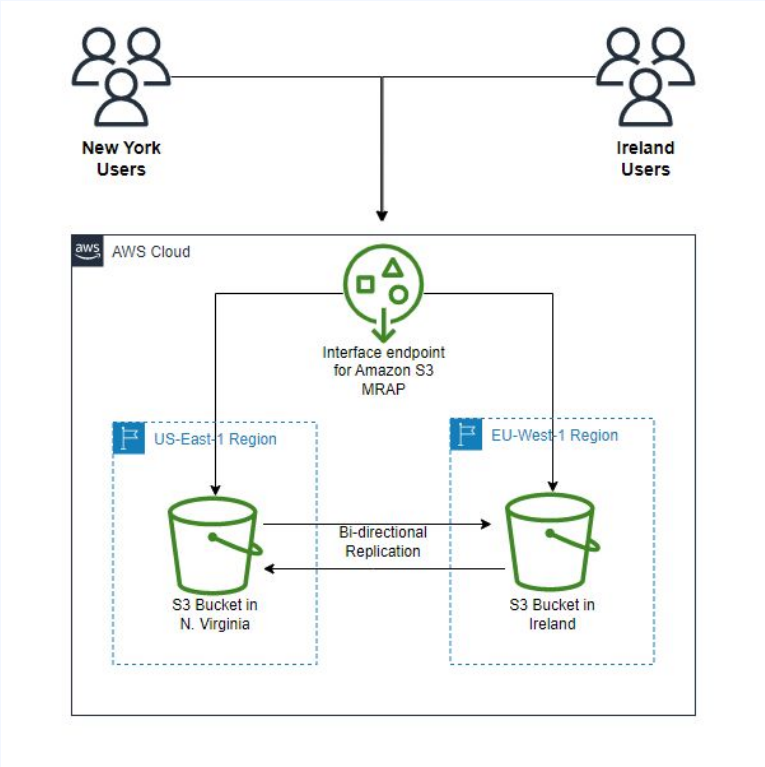
```
us_bucket = "mybucket-us.amazonaws.com"
eu_bucket = "mybucket-eu.amazonaws.com"
ap_bucket = "mybucket-ap.amazonaws.com"
default_bucket = "mydefaultbucket-us.amazonaws.com"

mapping = {
    "us-east-1": us_bucket,
    "us-east-2": us_bucket,
    "eu-central-1": eu_bucket,
    "eu-west-2": eu_bucket,
    "ap-northeast-1": ap_bucket,
    "ap-northeast-2": ap_bucket
}
```

How to achieve fast global response on low traffic files?

Amazon S3 Multi-Region Access Points

MRAP

- Automatically routes S3 requests to the closest region
- Great for low-hit or long-tail content
- Makes cache misses much cheaper
- Removes the cross-region penalty entirely



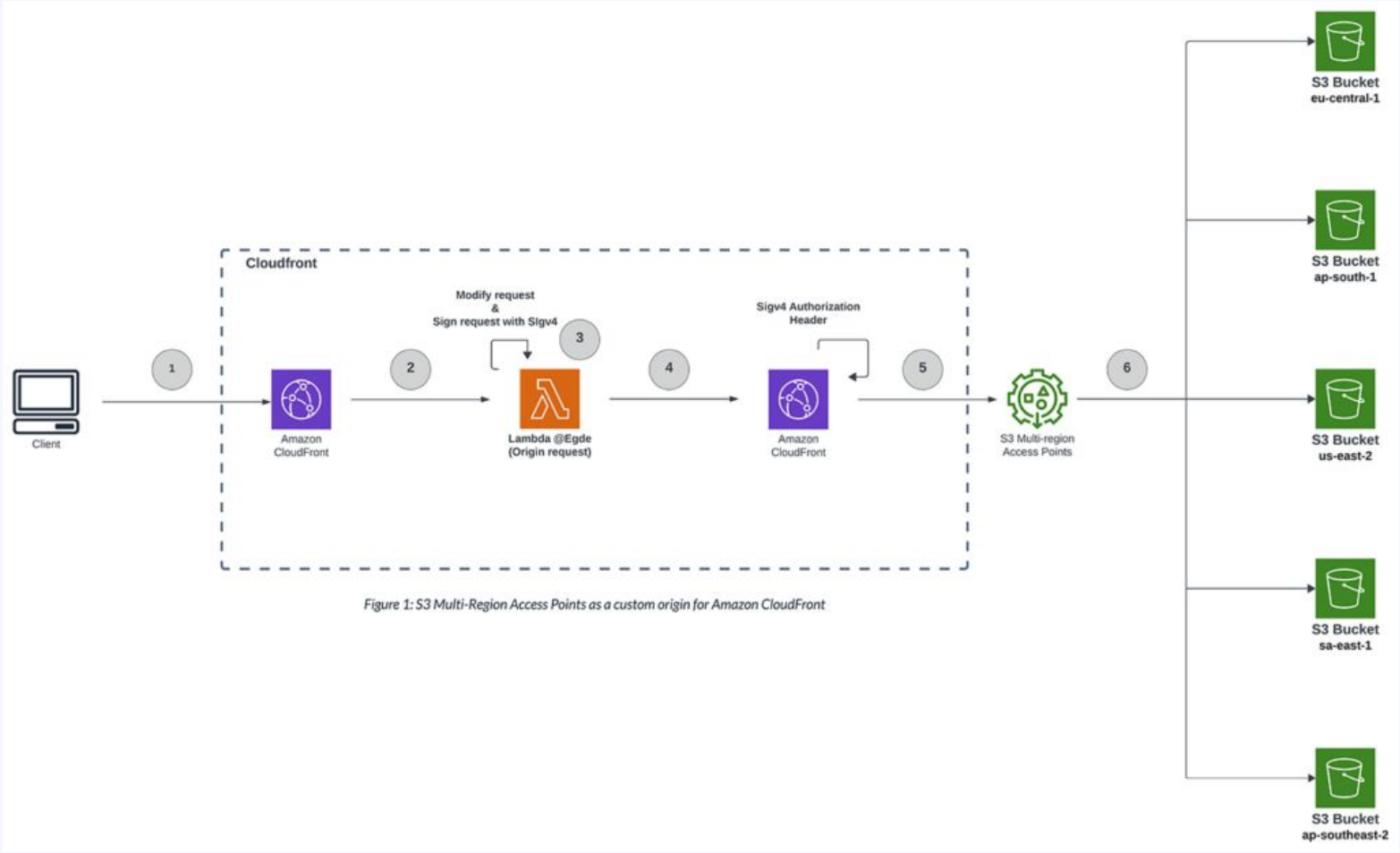
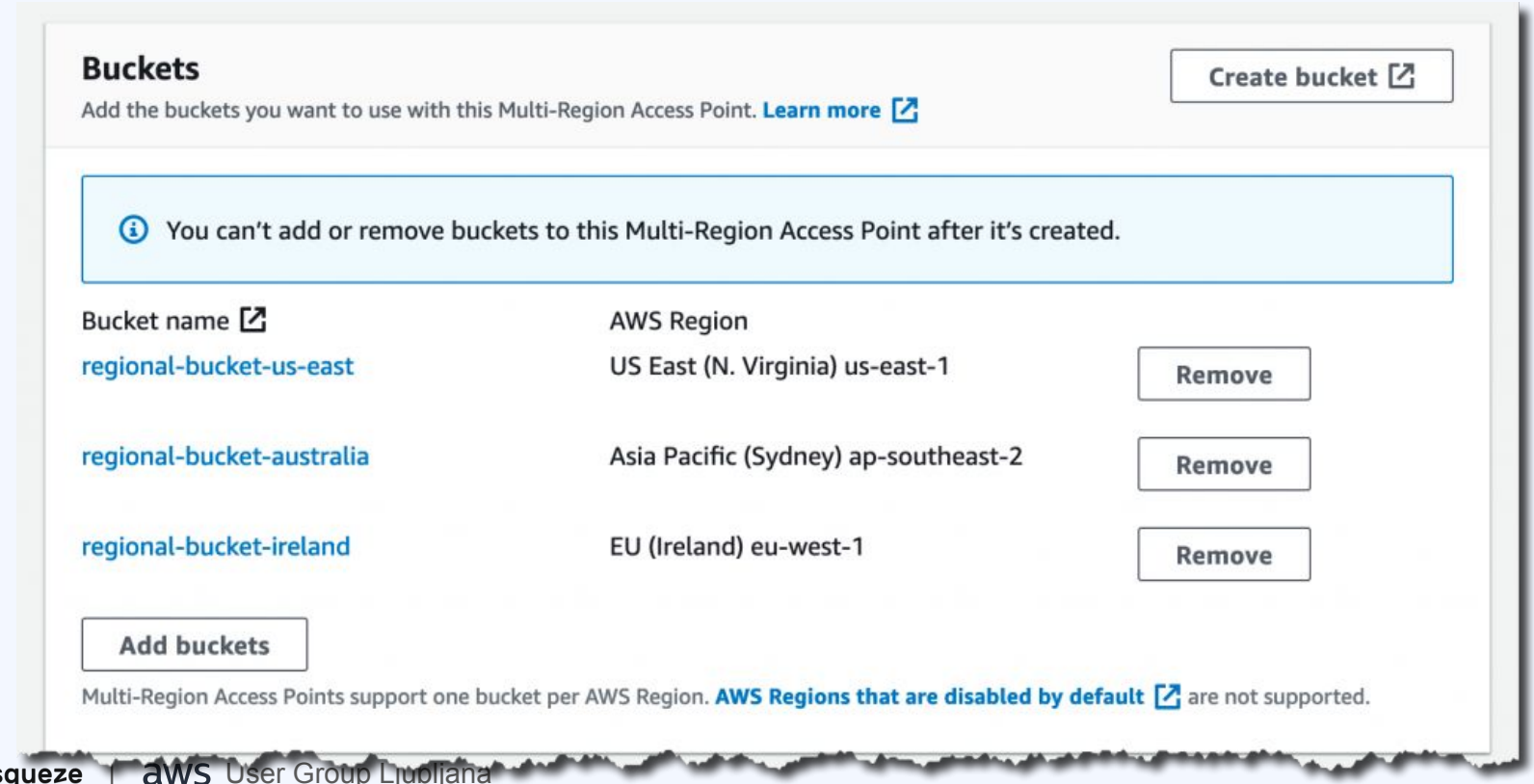
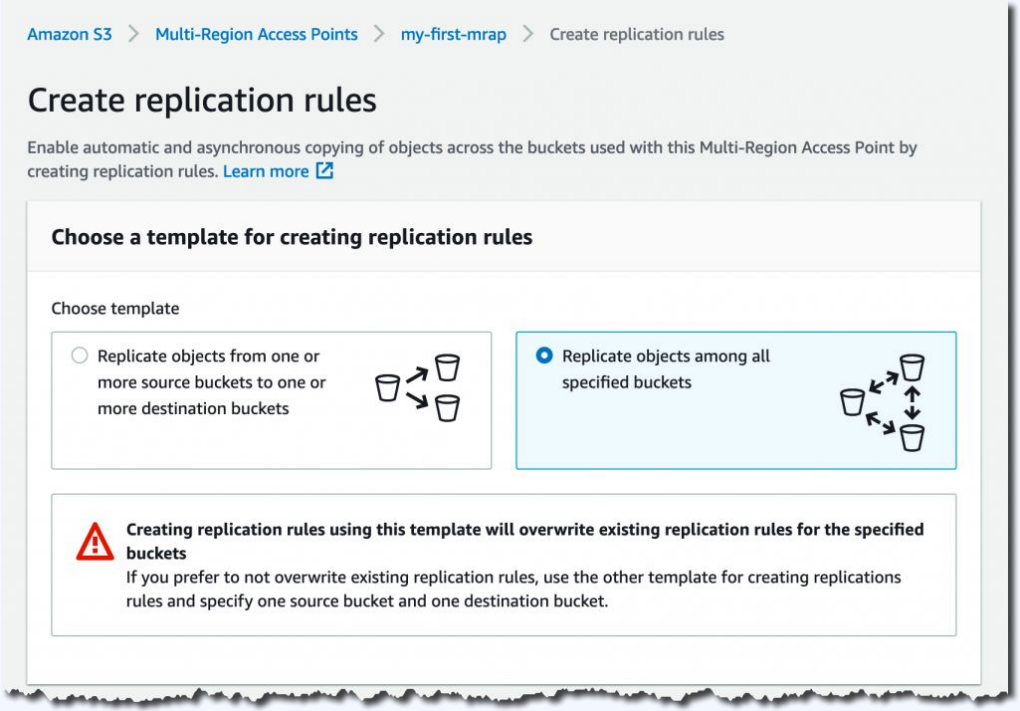


Figure 1: S3 Multi-Region Access Points as a custom origin for Amazon CloudFront

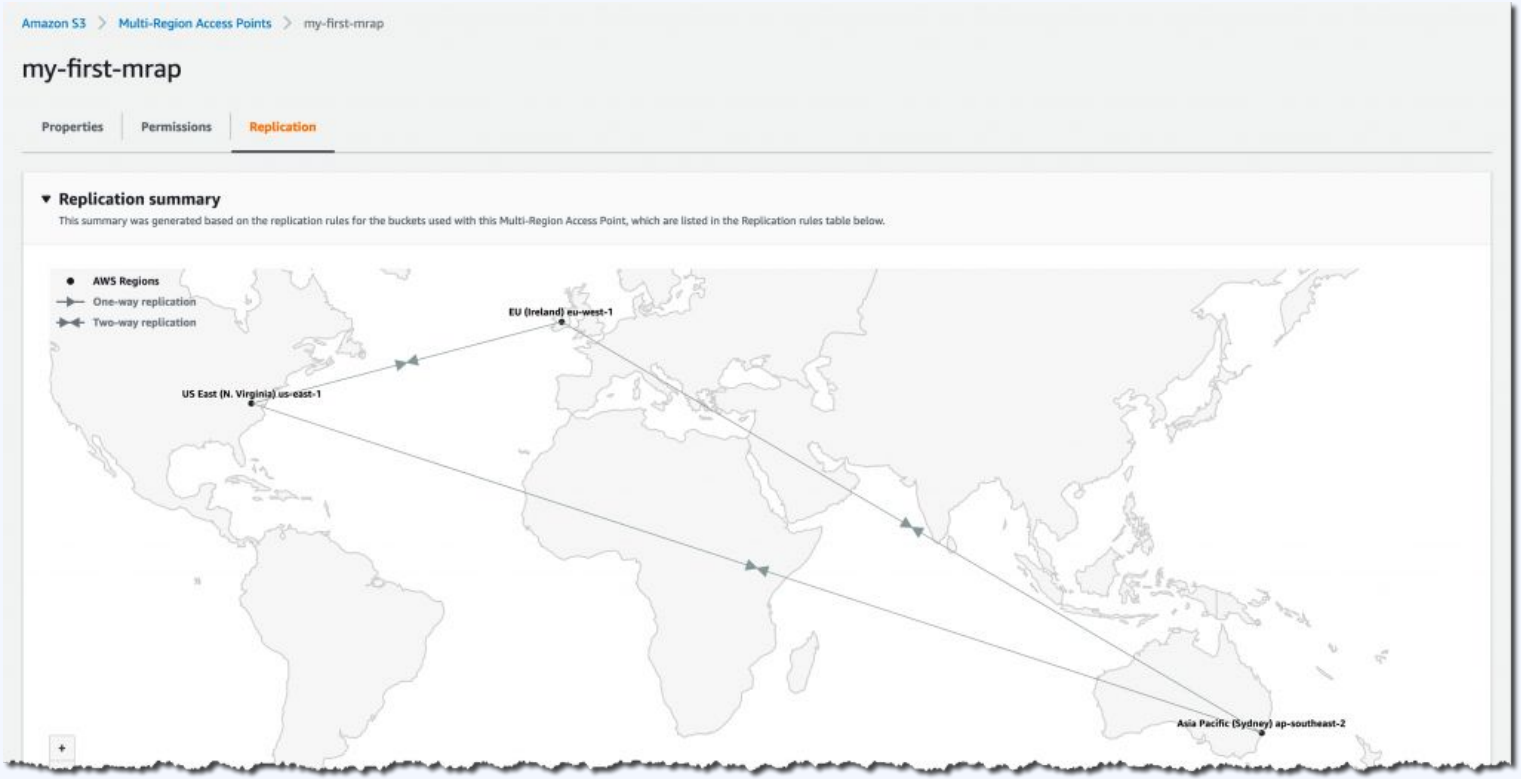
MRAP


Buckets
Add the buckets you want to use with this Multi-Region Access Point. Learn more
Create bucket
You can't add or remove buckets to this Multi-Region Access Point after it's created.
Bucket name
AWS Region
regional-bucket-us-east
US East (N. Virginia) us-east-1
Remove
regional-bucket-australia
Asia Pacific (Sydney) ap-southeast-2
Remove
regional-bucket-ireland
EU (Ireland) eu-west-1
Remove
Add buckets
Multi-Region Access Points support one bucket per AWS Region. AWS Regions that are disabled by default are not supported.

MRAP


Amazon S3 > Multi-Region Access Points > my-first-mrap > Create replication rules
Create replication rules
Enable automatic and asynchronous copying of objects across the buckets used with this Multi-Region Access Point by creating replication rules. Learn more
Choose a template for creating replication rules
Choose template
Replicate objects from one or more source buckets to one or more destination buckets
Replicate objects among all specified buckets
Creating replication rules using this template will overwrite existing replication rules for the specified buckets
If you prefer to not overwrite existing replication rules, use the other template for creating replications rules and specify one source bucket and one destination bucket.

MRAP


Amazon S3 > Multi-Region Access Points > my-first-mrap
my-first-mrap
Properties
Permissions
Replication
Replication summary
This summary was generated based on the replication rules for the buckets used with this Multi-Region Access Point, which are listed in the Replication rules table below.
AWS Regions
One-way replication
Two-way replication
EU (Ireland) eu-west-1
US East (N. Virginia) us-east-1
Asia Pacific (Sydney) ap-southeast-2

How do we use it?

Instead of
https://my -bucket-name.s3.amazonaws.com/...

we get MRAP endpoint:
https://123456.mrap.s3-global.amazonaws.com/...

For presigning instead of SigV4 use SigV4A.

Small performance note: Asymmetric cryptography is more computationally expensive than the HMAC algorithm that's used as the final step of SIGv4, but AWS handles the more expensive verification part.

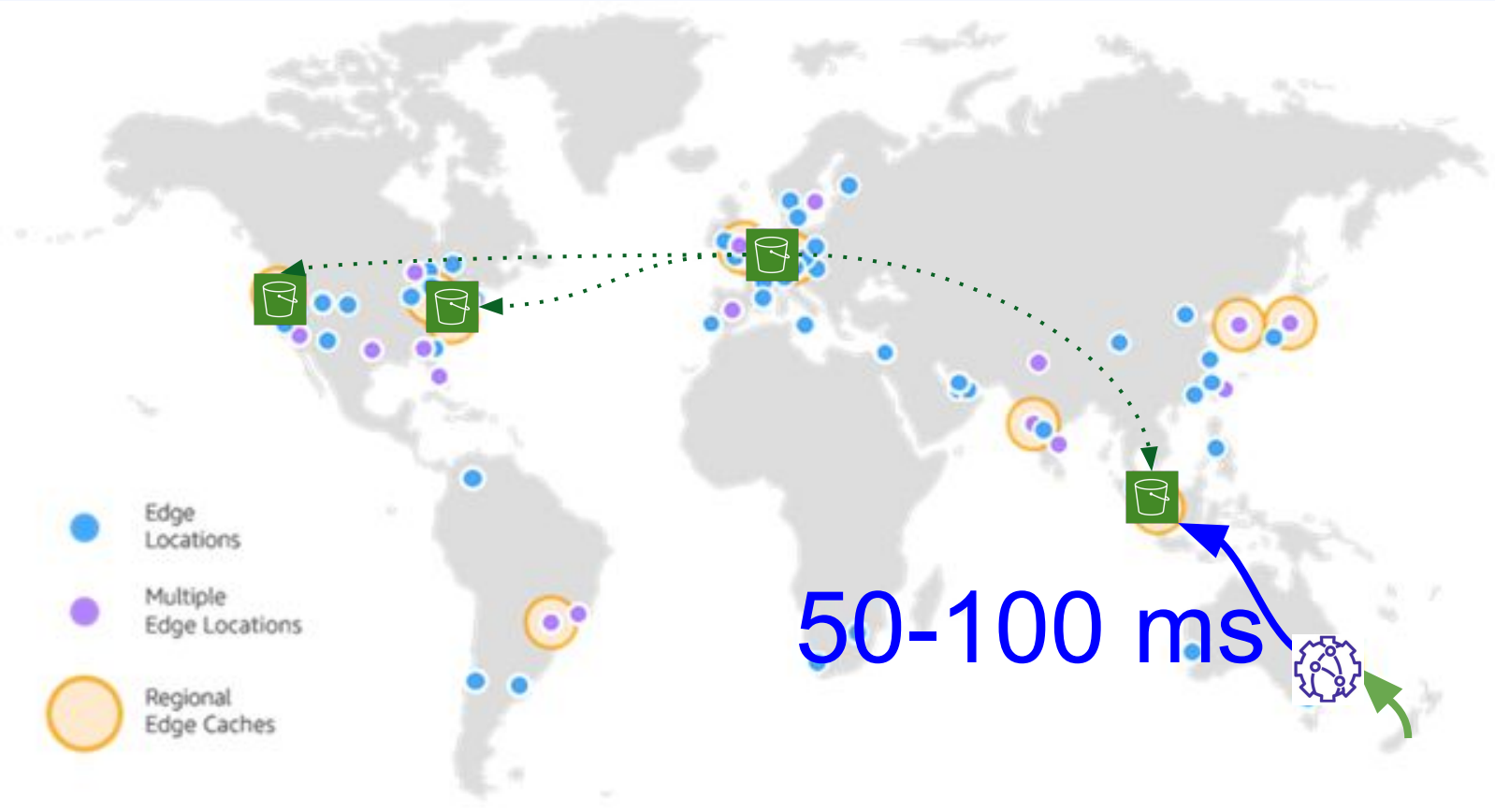
```
const minimalRequest = new HttpRequest({
  hostname: request.headers['host'][0].value,
  path    : encodeS3URI(decodeURIComponent(request.uri)),
  headers : headersMap,
});
const signer = new CrtSignerV4({
  service         : 's3',
  region          : '*',
  credentials     : defaultProvider(),
  sha256          : Hash.bind(null, 'sha256'),
  signingAlgorithm: AwsSigningAlgorithm.SigV4Asymmetric
});
const signedRequest = await signer.presign(minimalRequest,
{ signingService    : 's3',
  signingRegion     : '*',
  unhoistableHeaders: new Set(['x-amz-cf-id'])
});
```

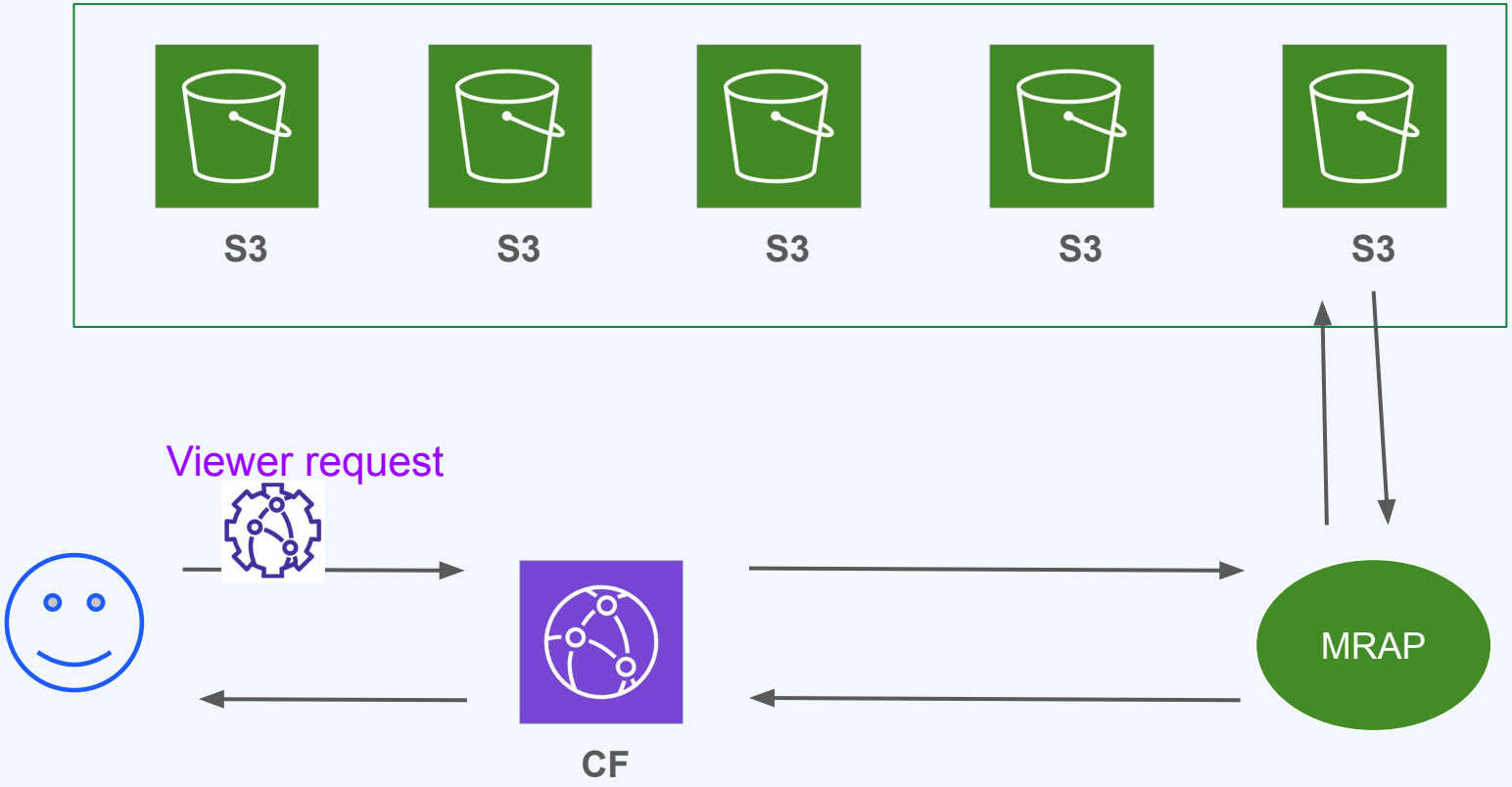
Signing process

There are multiple libraries out there that can do that, we preferred aws-sdk.

Good read: AWS SIGv4 and SIGv4A

How does cache miss look with MRAP implemented?




S3
S3
S3
S3
S3
Viewer request
MRAP
CF

What we gained

- Stable global response, even with cold caches
- ~100ms worst-case global latency on first load
- Zero dependency on cache warming
- Simple tenant-level routing
- Fewer moving parts, all AWS-native

One tip when debugging CloudFront

Always inspect the `x-cache` header

- Miss = expected for first request
- Hit = expected for follow-up requests, otherwise you might have configured something wrong

Posted on: Nov 20, 2025

Amazon CloudFront Functions now supports CBOR Web Tokens and Common Access Tokens

Bonus, did you know about Amazon CloudFront Embedded POPs

What they are

- CloudFront servers deployed inside local ISPs
- Act as mini edge locations closer to end users

Why they matter

- Shorter network path → faster first byte
- Lower latency even if you're far from AWS regions
- More stable performance during peak local traffic

Q&A