Building a fully Serverless AWS-based URL shortener using just a few services

9 June 2022 - 5 min. read

Amazon CloudFront *Amazon Route 53* *AWS Lambda@Edge* *Edge Computing* *Serverless*

Today is essential to fully exploit the potential of the cloud to carry out efficient and successful projects. The **serverless paradigm** is one of the most disruptive paradigms, still revolutionizing the sector.

By exploiting the principles of serverless computing, it is possible to build applications in which the infrastructural component **scales automatically**, perfectly following the demand curve. An excellent serverless infrastructure allows you to **reduce infrastructure and maintenance costs** by **reducing the customer's area of responsibility**. The entire infrastructure is **highly available**, scalable, and elastic by design.

In this article, we want to introduce you to a **completely serverless solution to create a URL shortener**. In the following paragraphs, we will show a real-world use case: from the requirements to the design of a solution based on AWS' fully managed and serverless services.

The requirements

The URL shortener provides an API that can be used for link creation.

The API should respond to a GET request like this one
https://example.com/api/v1/action/shorten?url=[URI]

The response must be plain text, and the body ought to contain the shortened URL
https://example.com/f/7427

When visiting the short URL, the URL shortener will redirect the visitor to the full URL.

No authorization nor authentication is needed for this version of the URL shortener.

The URL shortener should redirect and enforce HTTPS.

Our solution

We'll try not to use the obvious solution of just exposing a public RESTFul API using API Gateway, Lamba, and DynamoDB because it would not be fun :) *and* more importantly, it isn't the most cost-efficient solution.

We will take advantage of the following AWS services

- **Amazon S3** as Object storage for links metadata and as a redirection engine.
- **Amazon CloudFront** as CDN and single entry point for all requests.
- **AWS Lambda@Edge** for the backend computing required to generate the shortened link.
- **Amazon Route53** to manage DNS Entries.
- **AWS Amazon Certificate Manager** to generate and automatically renew HTTPS certificates.

The idea is to build or straightforward API leveraging Lambda@Edge.

Lambda@Edge is a way to use Amazon CloudFront as the entry point of your code and run the Lambda closer to users of your application. **This improves performance and reduces latency**; it also has built-in caching capabilities. Lambda@Edge allows you to build a really global application without providing or managing infrastructure in multiple locations worldwide.

When the user invokes the link shortener API, the request reaches the best CloudFront node for the user. CloudFront will then start a lambda in the nearest location.

The **Lambda logic should be very simple**; it basically computes **k,** a unique 8 characters string composed of uppercase letters, lowercase letters, and numbers. The total number of links that can be simultaneously active is, therefore, 218.340.105.584.896 (218+ Tera links).

We now need a place to store our unique key and the corresponding URL, and DynamoDB is the more obvious choice.

However, there is a feature of Amazon S3 that is almost unknown, which can be used to further decrease the cost of the solution, maintaining excellent performances and scalability.

Redirect requests for an S3 object

You can redirect requests for an object to another object or an arbitrary URL by setting the website redirect location in the object's metadata. Adding the x-amz-website-redirect-location property to S3 object metadata is all you need to obtain a 301 redirect each time you request the object key. In addition, the object can be empty (0 bytes).

We can take advantage of this feature by using S3 as a CloudFront enabled data store for our key-value database, creating 0 bytes objects named using **k** as a unique name, and setting the x-amz-website-redirect-location to the original URL.

So back to our scenario: The Lambda@edge stores an empty object into an Amazon S3 bucket. The object key is set to **k**, and the x-amz-website-redirect-location metadata is set to the original URL.

The objects contained in the S3 Bucket are used to cause the HTTP redirection when the user follows the link via CloudFront CDN.

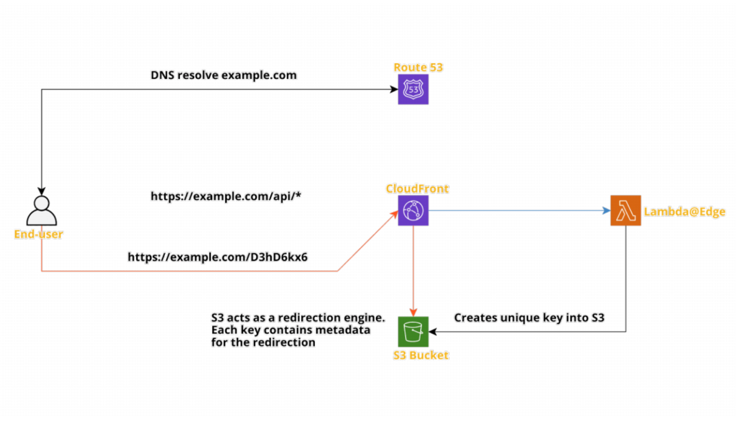
We need to **configure our CloudFront distribution to use the S3 Bucket as an origin**.

This solution is **inexpensive** and **fast**, thanks to the CloudFront distribution. It uses only serverless services and can scale fast and reliably to millions of impressions.

It is also possible to configure a mechanism to delete old links using S3 lifecycle policies.

AWS Infrastructure

The following is the infrastructural diagram of the solution.


DNS resolve example.com
Route 53
https://example.com/api/*
CloudFront
Lambda@Edge
End-user
https://example.com/D3hD6kx6
S3 acts as a redirection engine. Each key contains metadata for the redirection
Creates unique key into S3
S3 Bucket

The system's entry point is always CloudFront CDN; it integrates both Lambda@Edge for the API and Amazon S3 for the links redirects. The routing is path-based.

The CDN may or may not cache the API results; therefore, you can configure CloudFront to cache API responses, reducing the compute cost of the solution. If you enable caching, subsequent calls to shorten the same URL will get the same shortened version and not execute a Lambda function. If not, each request will trigger the Lambda and produce different and unique shortened links.

Note that for this solution, the VPC is not used at all; it leverages fully managed services in the AWS networking space.

Redirection Engine

The redirection engine is made using Amazon S3.

The Bucket is configured as public static web hosting, which the redirection engine requires. Each object has metadata to instruct the static web hosting to redirect the user with a 301 HTTP status code.

This Bucket is used as one of the origins for the CDN, so the actual content is served via CDN, not directly from S3. We can configure S3 so that the content can only be requested via CloudFront, reducing the risk of unwanted S3 access and related costs.

The redirect for each unique key is cached into the CDN for cost-saving and performance.

Logging and Debugging

The Lambda is configured to stream logs to AWS CloudWatch, and it is possible to set the retention period for each log group.

Due to the Lambda replication (Lambda@Edge) logs being streamed to the nearest AWS Region, logs are stored in the same region where the link is created.

Wrapping up

So to wrap up, this is a fully serverless, highly available, multi-region solution to build a simple URL shortener. This is a perfect starting point for further refining and discussions! Have you ever deployed something similar? Let us know in the comments!

Stay tuned for other solutions on Proud2beCloud!

Alessio Gandini

Cloud-native Development Line Manager @ beSharp, DevOps Engineer and AWS expert.Since I was still in the Alfa version, I'm a computer geek, a computer science-addicted, and passionate about electronics all-around.At this moment, I'm hanging out in the IoT world, also exploring the voice user experience, trying to do amazing (Io)Things.Passionate about cinema and great TV series consumer, Sunday videogamer

---