GO SERVERLESS! PART 1: LET'S CREATE A FILE SHARING APPLICATION BASED ON AWS SERVICES

AWS Lambda | CI/CD | Continuous Delivery | DevOps

beSharp | 11 September 2018

Go to part 2 | Go to part 3

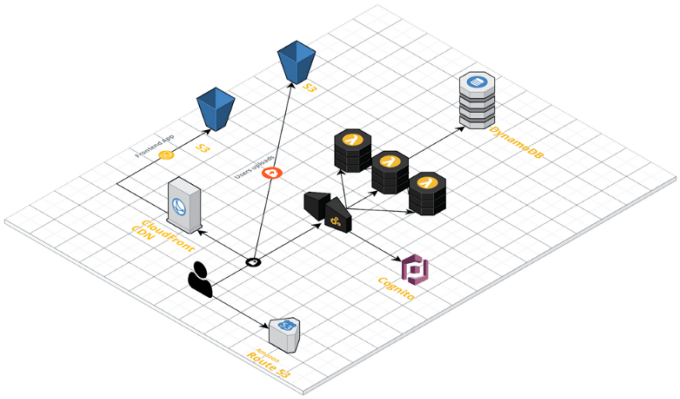
In recent years, the term **"Serverless"** has become more and more popular in the IT world.

Another Buzzword? Or is it indeed possible to develop an application that does not use any servers, as this word would suggest? Let's try to clarify this possible misunderstanding...

Serverless is a **Cloud paradigm** that allows applications to run independently from the underlying infrastructure. Thanks to Cloud services such as (for example) **AWS Lambda, API Gateway, and EKS**, it is possible to develop applications for which provisioning, scalability, and management operations are carried out transparently and automatically, without manual intervention from those who write the code.
This new technology has many advantages which improve the experiences of both user and developer. Knowing how to make the most of them is crucial to creating highly scalable, high-performance competitive solutions.

With this 3-article series, we will explain how to build a **file-sharing system** complete with login, "drag and drop" capability and file sharing using links, exploiting the potential of serverless technology and the services provided by AWS.

Amazon offers a wide range of **managed services** that can be used to implement completely serverless architecture rapidly: we will, therefore, create infrastructure as shown in the illustration below, then analyze in detail how to configure the necessary triggers for responding to events, such as file uploads.

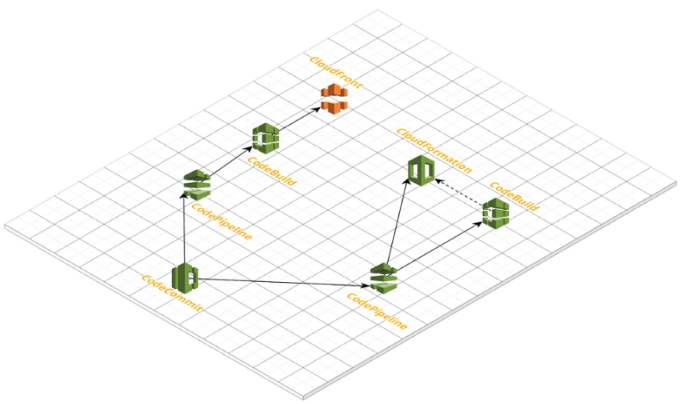


The application will consist of a static front-end and a range of back-end APIs, both protected by authentication. File information and sharing data will be saved to a key-value type database, while all files will be saved to object storage.

There is an interesting point worth noting: user uploads will be carried out directly to object storage through a specific authorization mechanism. This will allow the application to **scale rapidly** to face increasing requests, minimizing the resources to allocate in advance and their related costs.

In order to create this solution, we will employ some of the most interesting Serverless services, such as **S3** for the storage of objects, **DynamoDB** as a database, **Lambda** for the execution of back-end APIs, **API Gateway** to display the APIs, **CloudFront** as a CDN to serve the front-end, and **Cognito** for user sign-ins and authentication services.

We will also tackle the topic of automatic code deployment, creating a pipeline of **Continuous Deployment/Continuous Integration** that allows software modifications to be issued quickly and automatically.



To be specific, two separate pipelines will be created to handle the front and back-end CD/CI. The back-end one will use a CloudFormation template to carry out Lambda function provisioning and update the code, while the front-end one will copy assets to the appropriate bucket and manage the CDN with the necessary invalidations.

Let's start illustrating the services that are utilized according to design choices.

Amazon S3

"Amazon S3 is an object storage service created to memorize and restore any volume of data from any source. It is a service that provides extremely durable, readily available storage infrastructure with unlimited scalability, at a lower cost."

We will use S3 both as a source for files that make up the front-end application and as storage for user-uploaded files. Using this service allows for a virtually unlimited, highly-available space at a lower cost.

To make the implementation more secure, we will use two separate buckets to subdivide better permissions and keep application storage strictly isolated from user data storage.
The bucket reserved for static assets will be the source for the CDN utilized to distribute the front-end to clients.

CloudFront

"Amazon CloudFront is a global Content Delivery Network (CDN) that allows the distribution of data, videos, applications, and APIs to users with minimal latency and high transfer rates."

We will use CloudFront to serve the front-end of our application. In addition to improving performance and user experience, it also provides protection against **DDoS attacks** and reduces access costs to S3 for files that are frequently requested.

DynamoDB

"Amazon DynamoDB is a non-relational database that provides reliable performance on any scale. It is a multi-master, multi-region, fully-managed database that provides a constant latency of a few milliseconds, integrated security, a backup and restore service and a memory cache."

DynamoDB wins over other valid DBMS because the application saves metadata related to the uploaded files and their sharing settings, where everything is strong "file centric." The data structure fits well with the key-value model of DynamoDB, a condition which among other things is made possible by delegating the management of authorizations and user information to Cognito.

Furthermore, by choosing to use DynamoDB, the application benefits from a database in an almost fully automatic way. The database is **fully managed**, readily available, and easily and automatically

horizontally-scalable.

Cognito

"Amazon Cognito allows tools to be quickly and easily added for logging, access and Web app and mobile device access controls. Amazon Cognito allows resources to be recalibrated for millions of users and supports access with social identity providers like Facebook, Google and Amazon and corporate identity providers via SAML 2.0."

We chose to delegate authentication and user data management to a managed service. This allows us to limit the attack perimeter of the service we're building, increase login and authentication possibilities, and provide a fast, bulletproof integration with API Gateway. The latter allows us to reduce costs because all the effort of authentication is carried out by the gateway and does not affect the Lambdas' calculation time. Furthermore, any incorrect login attempts or calls made with expired or incorrect tokens will never reach the application but will be handled directly on the architecture's outer perimeter.

The managed user data service also allows us to eliminate database traffic to find key information about users. It **minimizes maintenance**, the securing of tables with sensitive data, and credential management.

Lambda

"AWS Lambda allows code to be run without having to manage servers or carry out provisioning. Rates are calculated based on processing times, so no charges are made when the code is not running."

The back-end will be developed entirely with AWS Lambda. This technology—key to the serverless philosophy—allows us to pay for single operations and only for the time it is actually used, eliminating wastage of processing power and costs when the system is idle. It also saves us from having to configure and test an adequate auto-scaling policy.

API Gateway

"Amazon API Gateway is a fully managed service that simplifies for developers the creation, publication, maintenance, monitoring, and protection of APIs on any scale."

In this case, it's an obvious choice; only AWS API Gateway has all the hooks and integrations necessary for seamless and effective assimilation into the application architecture. The main goal is to provide a readily available, high performance, reliable interface for back-end lambda functions. It also allows integration with Cognito for authenticating calls and passing them on to the back-end only if they are correctly authenticated and legitimate, effectively blocking all invalid requests. Another advantage is that we can easily integrate CloudFront to cache the back-end responses in order to **reduce computing times** and, at the same time, increase application responsiveness by reducing latency.

In the next articles, we will take a detailed look at the application structure, serverless infrastructure, and the triggers to configure in order to respond to events significant to the application.

Stay tuned!

Go to part 2 | Go to part 3

beSharp

Dal 2011 beSharp guida le aziende italiane sul Cloud. Dalla piccola impresa alla grande multinazionale, dal manifatturiero al terziario avanzato, aiutiamo le realtà più all'avanguardia a realizzare progetti innovativi in campo IT.

Get in touch

beSharp.it
proud2becloud@besharp.it

Copyright © 2011-2021 by beSharp srl - P.IVA IT02415160189