AWS CLOUDFORMATION: THE “TOP 5” REASONS FOR NOT USING IT

AWS CloudFormation

beSharp | 3 May 2019

How to survive the most common pitfalls and take advantage of this service, without too many headaches.

Who hasn’t had problems with **AWS CloudFormation?**

Sadly, it is commonly known that AWS CloudFormation is as powerful of a tool as it is frustrating; this short article will highlight the most common pitfalls and the most annoying situations, as well as, obviously, **ways to avoid them.**

We begin our journey by **describing the service:**

“AWS CloudFormation provides you with a common language for describing and provisioning all infrastructure resources in your cloud environment. With CloudFormation you can use a simple text file to model and implement provisioning, in an automated and secure manner, of all resources needed for your applications on all regions and all accounts. Such file will be the only source of your cloud environment.

CloudFormation is available at no additional cost; you are only charged the cost of AWS resources needed to run the applications.”

One of the main advantages of using AWS CloudFormation is precisely to have **a template of your own infrastructure, which can be versioned, revised, and easily replicated** as often as necessary.

There are problems, however, right from the description of the service.

#1 An (in)complete modeling

Although the AWS service presentation page explicitly states that AWS CloudFormation offers complete cloud infrastructure modeling, this, in fact, does not happen.

At least, not always; in fact, **there are many configurations and/or resources that cannot be specified in an AWS CloudFormation template.**

To date, for example, it is not possible to:

- Define the settings related to Cognito UserPool for federation with an external IDP
- Add a route to a routing table that points to a Transit Gateway
- Implement SSH key provisioning for EC2 (probably for security reasons)

Our advice is to make sure that all the necessary services and configurations can be truly modeled with the current version.

Almost all services are supported, although for some it is necessary to wait a considerable amount of time between the release and official support of AWS CloudFormation.

There is also a way to **extend the capabilities of AWS CloudFormation**using custom resources. In fact, they allow the deployment of a Lambda function and execute it as a “Custom Resource” at a given time during the creation of the stack.

Here is the official documentation of this feature.

#2 Waiting time

If it is true that once the template is packaged, the execution is faster than the manual one, the template production process is a long process based on trial and error.

It is not possible to 100% check a template locally, it is only possible to be certain of syntactic correctness without first executing it.

Which means that prior to arriving at a working version, you have to go through numerous attempts, each having long execution times.

To make life easier we can count on the following features:

The first is the ValidateTemplate API which does a much more thorough check than just a linter.

Another way is the extensive use of changesets, which allow you to have a preview of the actions that will result in changes to a template.

It is also advisable to **proceed by making small and rapid changes: f**irst, it is best to create all the resources that require a low provisioning time, then, only at the end, add the most time-consuming resources to the stack, such as EC2 instances, RDS and the like. This will allow all resources' waiting and validation times that are probably required by provisioning services to be shorter.

#3 Unintelligible errors

Error messages are often meaningless, misleading, useless, or simply wrong.

One can only be certain that the stack has failed. **What has actually failed and why is as useful to know as it is difficult to find out.**

The only way to quickly find the reason for failure is to **analyze the execution log** to see exactly on which resource an error is found, and when the provisioning for that resource occurred.

#4 Cloudformation drift detection

AWS CloudFormation's drift detection was loudly requested by many users, and consists of the ability to **automatically detect if changes were made to the configuration of the stack resources outside CloudFormation** via the AWS management console, the CLI and the SDK

This is a very useful feature, unfortunately, in practice, it gives many false positives. In general, it indicates that it is likely that something has been modified externally and, therefore, that the stack can no longer be updated automatically. **Unfortunately, it does not indicate that the stack is not actually editable.**

It is best not to consider it as a reliable indication, at least for now.

#5 Hard limit to 200 resources per stack

It is not possible to create more than 200 resources in a single stack.

Many argue that this is not a problem at all and that it is easily circumvented with common sense. After all, why should you create more than 200 things at a time?

The problem lies in what is defined as a "resource"; in fact, there are many objects that are normally invisible while creating resources from the web console. When creating a simple infrastructure with a dozen lambdas, an API Gateway with relative policies, and some additional resources, **the hard limit can be easily reached.**

The microservices should be small, and a complex application should, therefore, consist of multiple CloudFormation stacks… however, even having many stacks becomes burdensome from an organizational and management point of view, thus **it is better to measure the amount of resources to be included in a single template** well, and to develop a strategy to partition a complex architecture.

These are **the 'top 5' reasons** for not using AWS Cloudformation. Well, at least these are five reasons not to use it regardless of the belief that it can solve everything without creating headaches.

In conclusion, AWS CloudFormation is powerful but insidious, better to plan ahead when starting a new project in order to avoid the most common mistakes.

Tell us your experience, comment with the most insidious error you have detected. The best stories will be featured in a forthcoming article on AWS CloudFormation pitfalls.

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