CloudFormation vs. Terraform

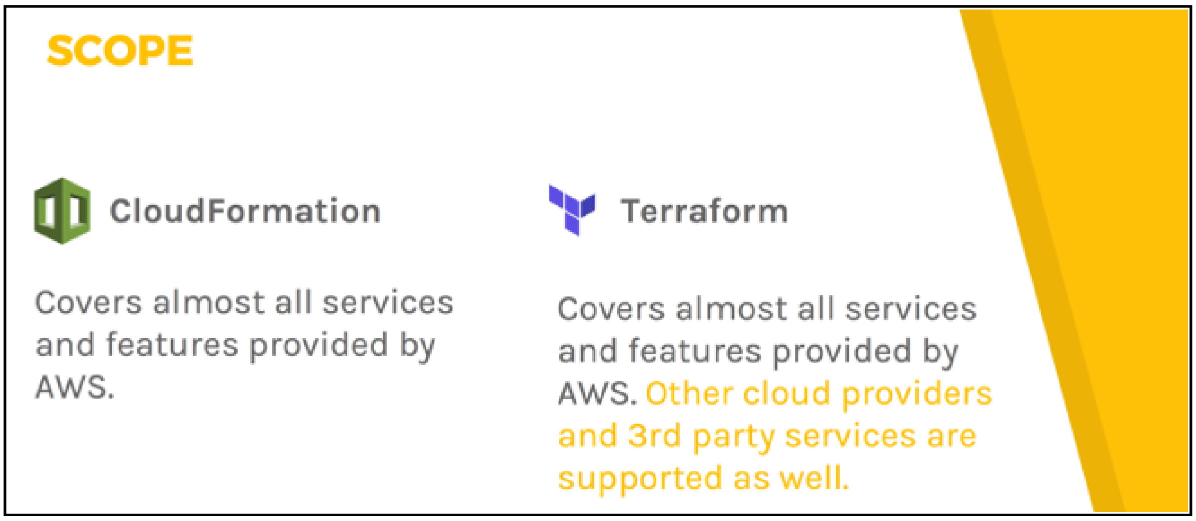
The most reliable way to automate creating, updating and deleting your cloud resources is to describe the target state of your infrastructure and use a tool to apply it to the current state of your infrastructure (see Understanding Infrastructure as Code). CloudFormation and Terraform are the most valuable tools to implement Infrastructure as Code on AWS. You will learn about the differences between CloudFormation and Terraform during this article.

Before we start, both tools are following a very similar approach.

1. You define a template (CloudFormation) or configuration (Terraform) describing the target state of your infrastructure.
2. The tool (CloudFormation or Terraform) calculates the necessary steps to reach the defined target.
3. The tool (CloudFormation or Terraform) executes the changes.

Scope

CloudFormation covers almost all bits and parts of AWS. Terraform covers the most important AWS resources as well. But on top of that Terraform can provision infrastructure at other cloud providers as well as 3rd party services. A shortened list of vendors supported by Terraform: Bitbucket, Datadog, DigitalOcean, Fastly, GitHubm, Gitlab, Google Cloud, Heroku, OpenStack, Mailgun, Microsoft Azure, and New Relic. Depending on your infrastructure a big or at least small plus for Terraform.



SCOPE

CloudFormation

Covers almost all services and features provided by AWS.

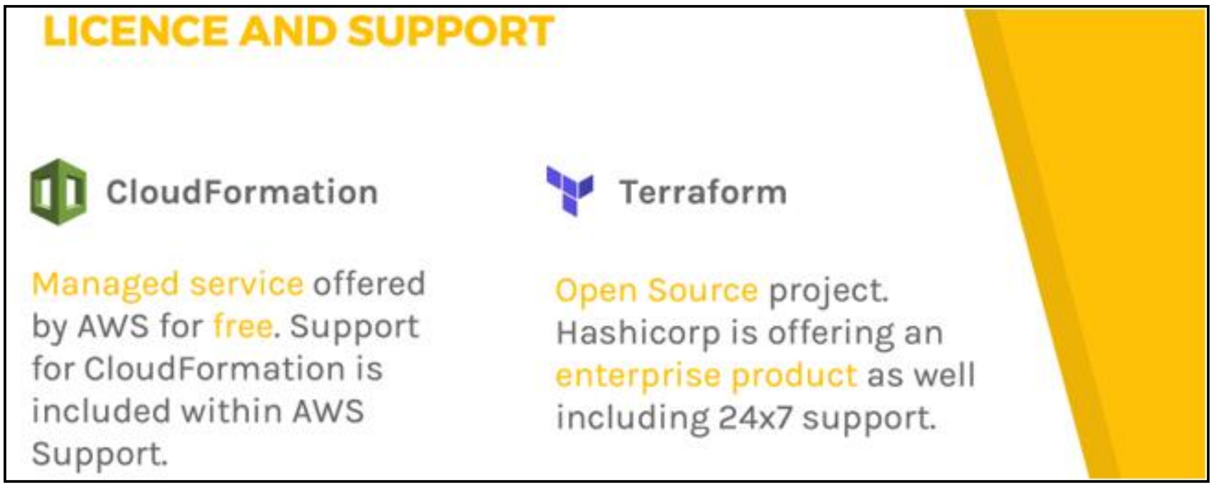
Terraform

Covers almost all services and features provided by AWS. Other cloud providers and 3rd party services are supported as well.

Licence & Support

CloudFormation is a service offered by AWS for free. The AWS support plans include support for CloudFormation. **Terraform** is an Open Source project. Hashicorp, the company behind Terraform, is offering support plans as well.

When already subscribed to an AWS support plan, that might be a plus for CloudFormation. If you prefer Open Source that is a plus for Terraform.



LICENCE AND SUPPORT

CloudFormation

Managed service offered by AWS for free. Support for CloudFormation is included within AWS Support.

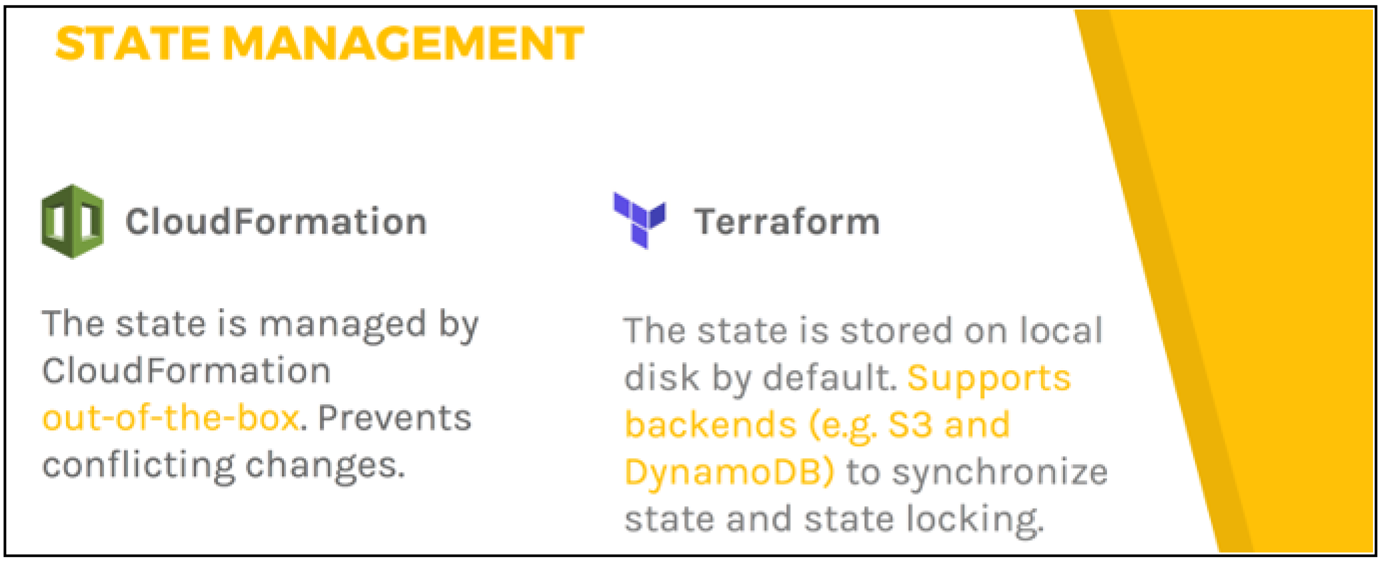
Terraform

Open Source project. Hashicorp is offering an enterprise product as well including 24x7 support.

State Management

Both tools need to keep track of all the resources under management. CloudFormation is managing its state with so called stacks. By default, Terraform is storing its state on disk. Terraform is offering remote state as well, for example, based on S3 and DynamoDB. It is advisable to use remote state when multiple users are working on the same infrastructure in parallel.

CloudFormation manages state within the managed service out-of-the-box which is a small plus compared to Terraform where you need to configure remote state yourself.



STATE MANAGEMENT

CloudFormation

The state is managed by CloudFormation out-of-the-box. Prevents conflicting changes.

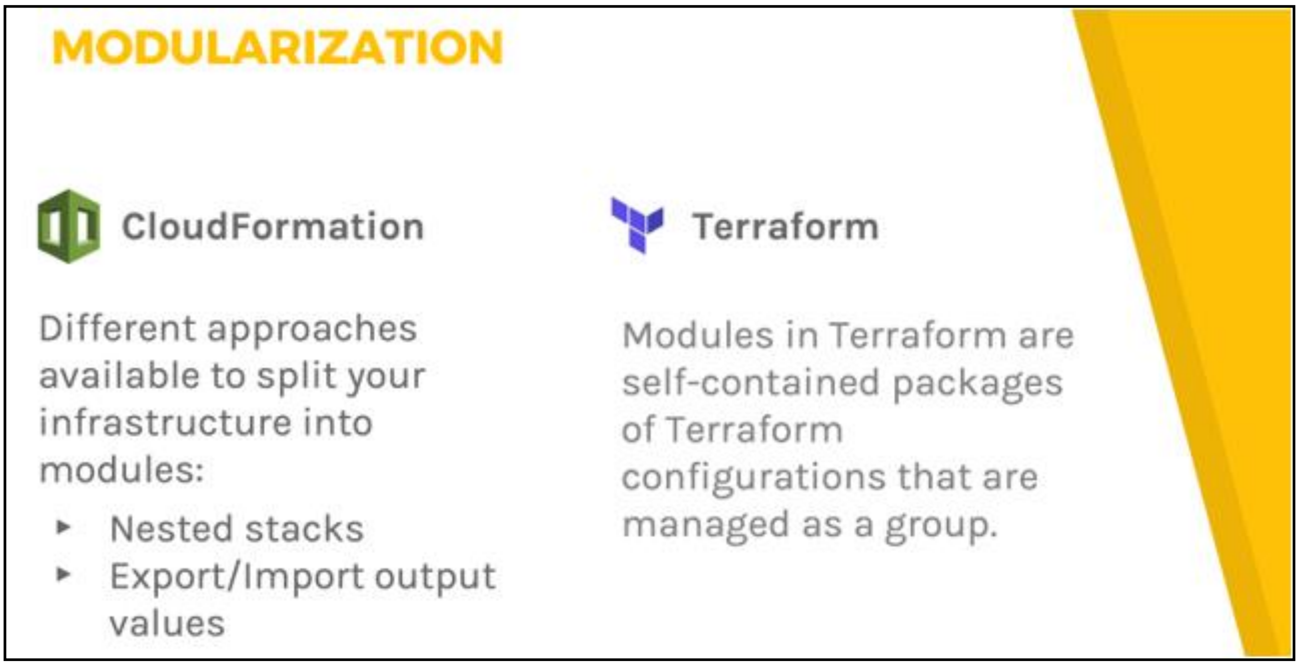
Terraform

The state is stored on local disk by default. Supports backends (e.g. S3 and DynamoDB) to synchronize state and state locking.

Modularization

An infrastructure for a typical web application consists of a lot of resources: VPC, Subnets, Security Groups, Auto Scaling Group, Elastic Load Balancer, to name a few. Specifying all these resources in a single blueprint will cause you headaches when maintaining the system in the future. Using small modules that you stick together as needed is a common approach.

Both CloudFormation and Terraform do support modularizing your infrastructure blueprints.



MODULARIZATION

CloudFormation

Different approaches available to split your infrastructure into modules:

- Nested stacks
- Export/Import output values

Terraform

Modules in Terraform are self-contained packages of Terraform configurations that are managed as a group.

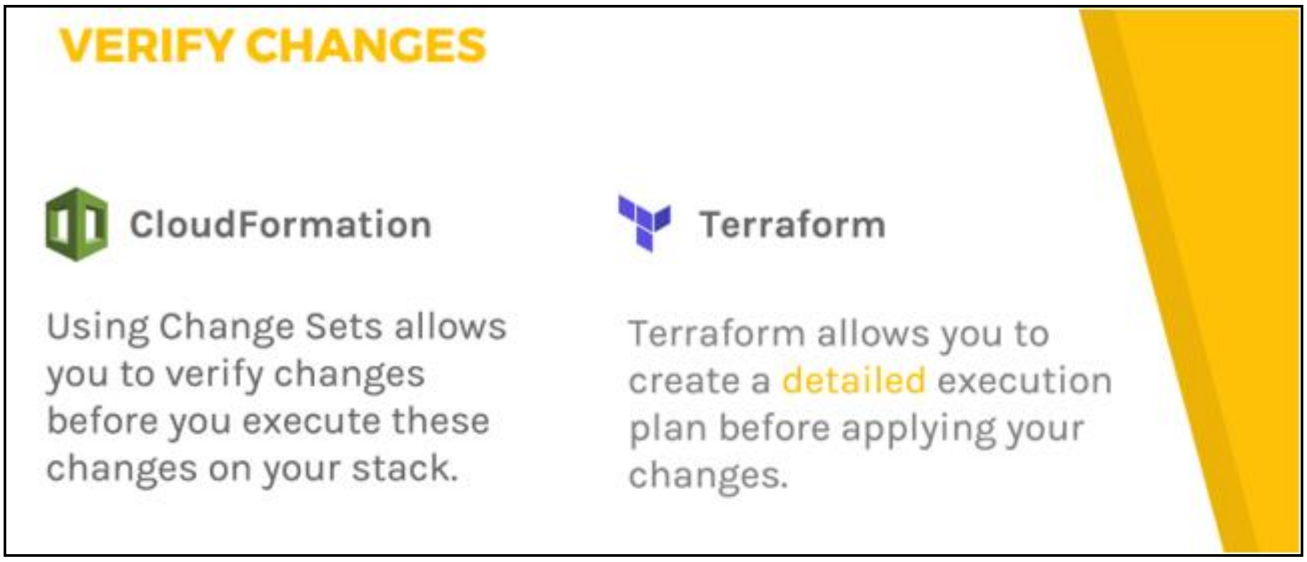
Handling modules with Terraform is simple. CloudFormation is offering multiple ways to create modules with different pros and cons. I’d award Terraform with a plus for usability.

Verify Changes

CloudFormation and Terraform do not only allow you to create your infrastructure from scratch automatically. You can use both tools to update your infrastructure later as well. When applying a change to your infrastructure based on changes you made to your blueprint it is soothing to be able to verify the changes the tool will execute on your infrastructure.

CloudFormation offers change sets that you can use to verify changes. Terraform provides a command named plan which gives you a very detailed overview of what will be modified if you apply your blueprint.

Terraform presents a detailed and readable summary of the changes that will be applied. That's a big plus compared to the basic overview CloudFormation is providing with a change set.



VERIFY CHANGES

CloudFormation

Using Change Sets allows you to verify changes before you execute these changes on your stack.

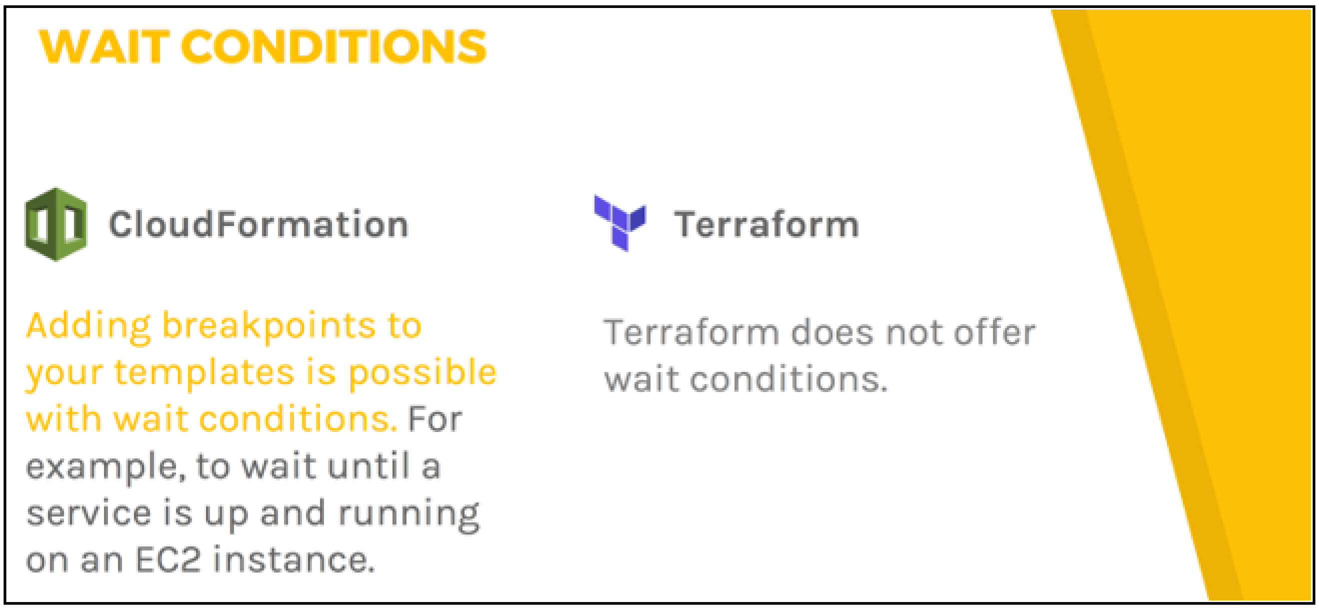
Terraform

Terraform allows you to create a detailed execution plan before applying your changes.

Wait Conditions

It is useful to be able to add wait conditions to your infrastructure automation from time to time. For example, if you want to wait until a service has been started on a virtual machine. Using wait conditions allows you to wait for signals sent via HTTPS before depended resources are created or updated when using CloudFormation.

Being able to use wait conditions is a plus for CloudFormation. Terraform does not support wait conditions.



WAIT CONDITIONS

CloudFormation

Adding breakpoints to your templates is possible with wait conditions. For example, to wait until a service is up and running on an EC2 instance.

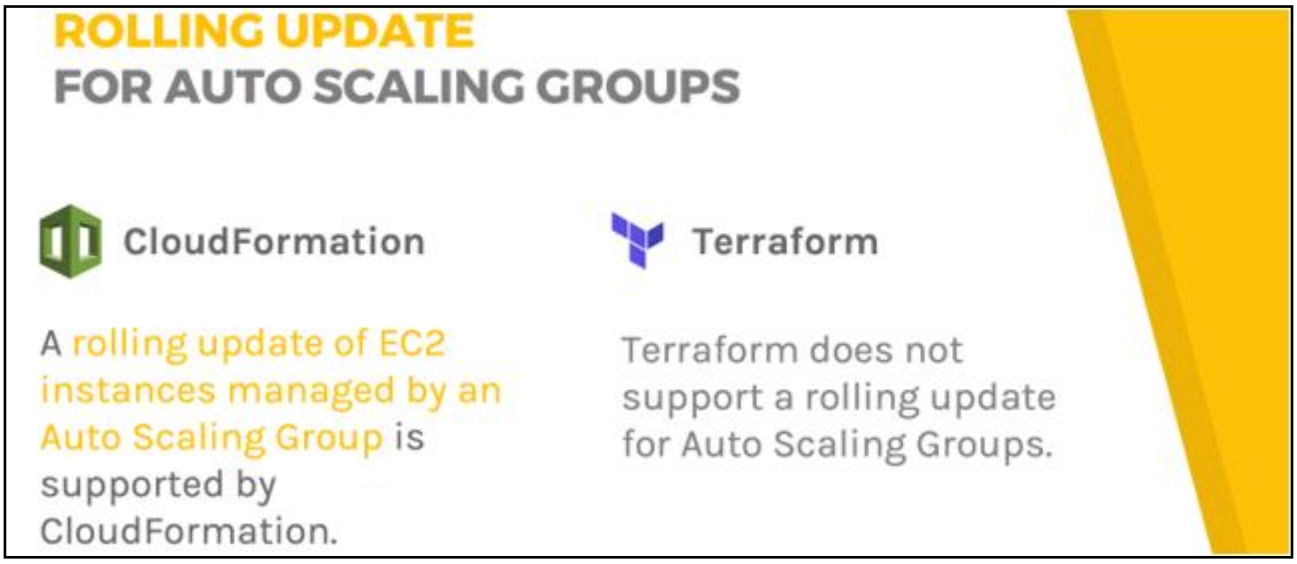
Terraform

Terraform does not offer wait conditions.

Rolling Update

What happens when you change a Launch Configuration of an Auto Scaling Group within your blueprint? When using an update policy, CloudFormation will perform a rolling update including a rollback in case of a failure. Terraform does not support rolling updates for Auto Scaling Groups out-of-the-box.

Supporting rolling updates for Auto Scaling Groups is a plus for CloudFormation.



ROLLING UPDATE
FOR AUTO SCALING GROUPS

CloudFormation

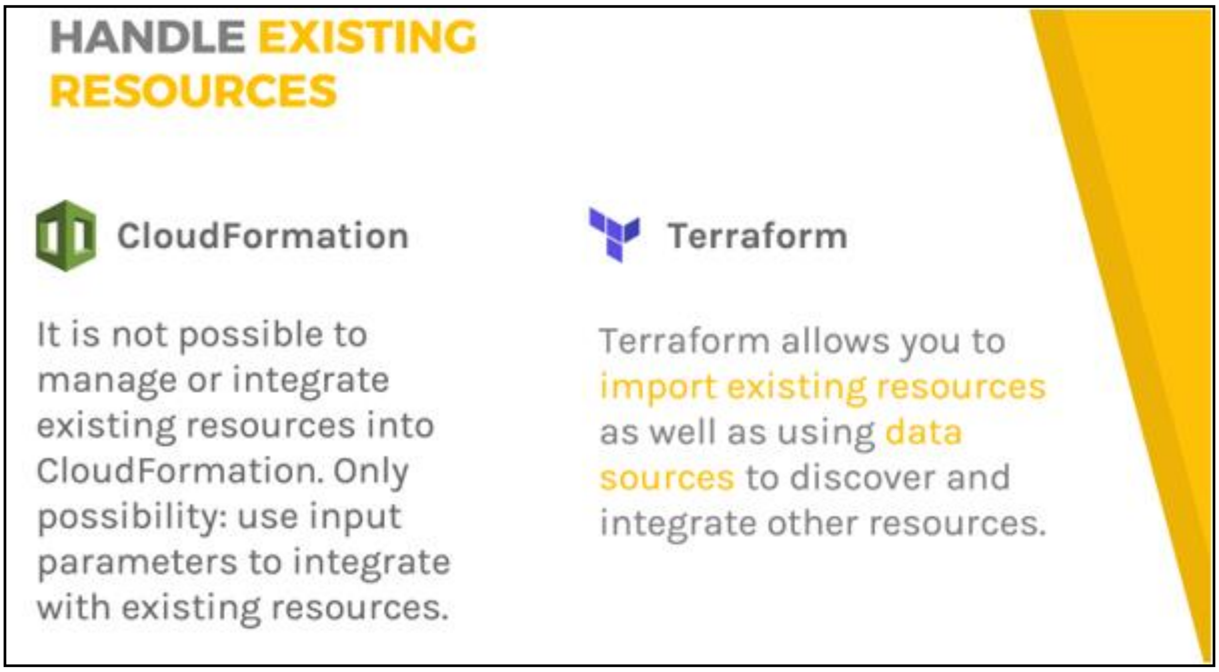
A rolling update of EC2 instances managed by an Auto Scaling Group is supported by CloudFormation.

Terraform

Terraform does not support a rolling update for Auto Scaling Groups.

Handle Existing Resources

Besides using input parameters, there’s no way to access existing resources with CloudFormation. It is also not possible to import an existing resource into a CloudFormation stack. Terraform allows you to import existing resources. On top of that data, providers enable you to query attributes from existing resources. Handling existing resources is a plus for Terraform.



HANDLE EXISTING RESOURCES

CloudFormation

It is not possible to manage or integrate existing resources into CloudFormation. Only possibility: use input parameters to integrate with existing resources.

Terraform

Terraform allows you to import existing resources as well as using data sources to discover and integrate other resources.

Summary

CloudFormation and Terraform are both powerful and mature tools. Going through the differences listed above will help you to make a decision for CloudFormation or Terraform depending on your use case. Personally, I prefer CloudFormation when the evaluation based on the differences of the tools does not produce a clear winner.

Source: https://cloudonaut.io/cloudformation-vs-terraform/