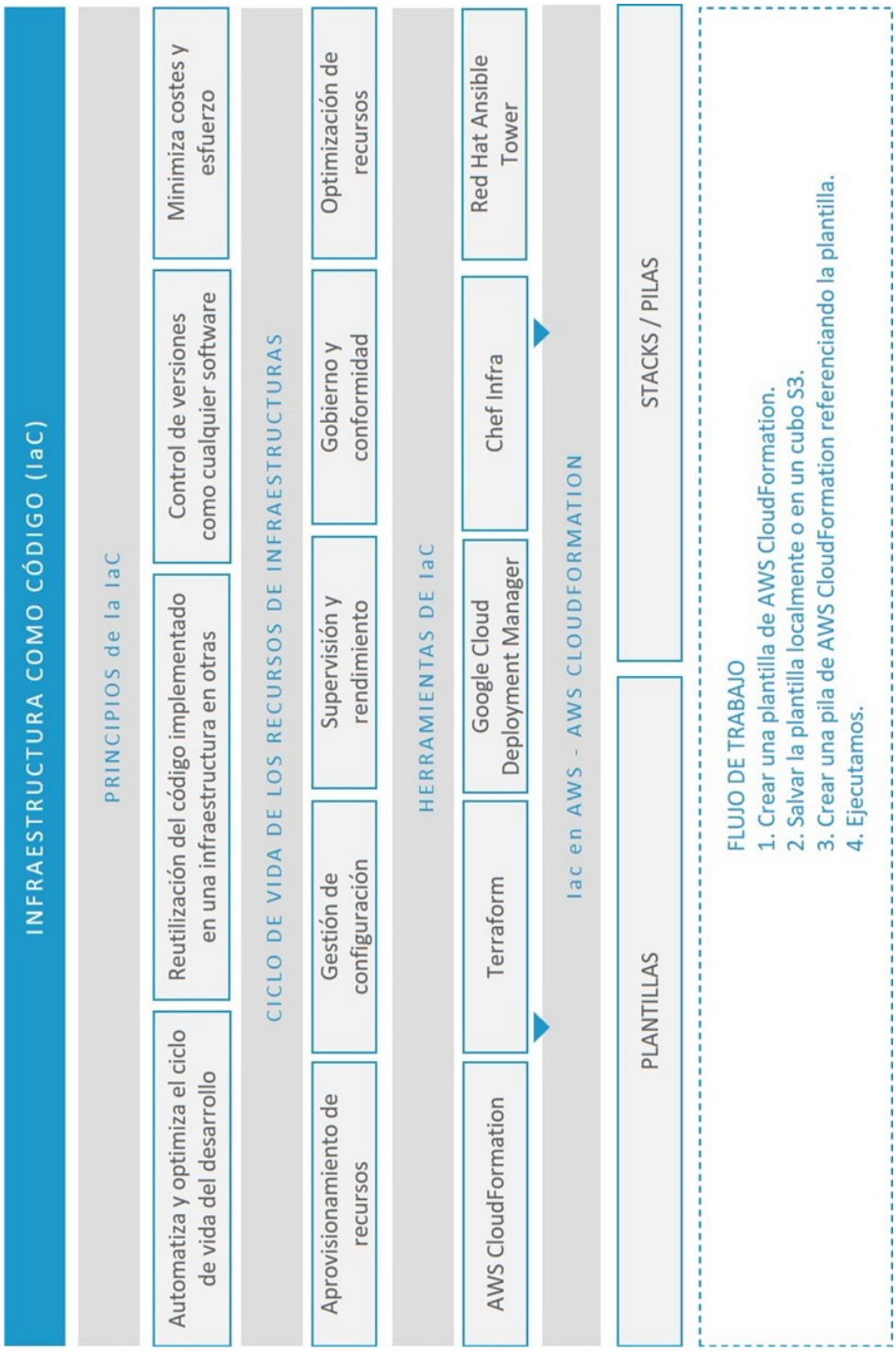
Cloud Computing, DevOps y DevOps Culture

Tema 8. Infraestructura como código

Índice

Esquema



8.1. Introducción y objetivos

Hasta hace unos años, los responsables de sistemas tenían que montar sus infraestructuras de hardware y software de forma manual. Es decir, que montaban y apilaban el hardware para posteriormente instalar y configurar los sistemas operativos y aplicaciones necesarias para el negocio. Como puedes imaginar, este proceso, además de ser muy tedioso, normalmente se ejecutaba por más de una persona, lo que generaba una larga espera hasta que la infraestructura de aplicaciones y sistemas estuviese disponible. Para agilizar estas implementaciones, optimizar el rendimiento y entregar con más rapidez el software desplegado en las plataformas, los responsables de sistemas crearon scripts (pequeñas porciones de código a bajo nivel) para unificar, automatizar y optimizar el proceso.

A su vez, la aparición de las metodologías ágiles ha instaurado una nueva forma de gestionar el software que implica que la instalación, configuración y mantenimiento de los sistemas informáticos se haga más compleja, si es posible. Estas metodologías requieren de varios ambientes donde ejecutar las aplicaciones (desarrollo, *testing*, *staging*, etc.), lo cual termina duplicando, al menos, el esfuerzo de los administradores.

La respuesta que ha dado la informática a estos problemas es la optimización y automatización del despliegue de aplicaciones y la configuración de los servidores. Actualmente, es posible configurar servidores a través de la programación y sin intervención de los administradores. Esta nueva forma de administración, que considera a la infraestructura como un aplicativo más a ser gestionado de manera análoga al software, se la conoce **como infraestructura programable o infraestructura como código,** más conocida como IaC (Huttermann, 2012).

Ideas clave

En este tema se pretenden conseguir los siguientes objetivos:

- Conocer qué es la Infraestructura como código.
- Analizar el ciclo de vida de los recursos de infraestructura.
- Conocer cuáles son las herramientas de la Infraestructura como código.
- Analizar cómo se implementa la infraestructura como código en AWS.

8.2. ¿Qué es la IaC?

En 2011, Andressen Marc escribió un artículo llamado «Why Software is Eating the World». La idea central de este artículo está en que cualquier proceso que puede ser programado, se convierte en software. Esto se ha convertido en una especie de abreviatura para las tesis de inversión detrás de la actual ola de *startups* unicornio de Silicon Valley. También es una idea unificadora detrás del conjunto más amplio de tendencias tecnológicas que vemos hoy en día como *machine learning*, IoT, *ubiquitous mobile connectivity*, SaaS, y Cloud Computing. Estas tendencias están haciendo que el software sea más abundante y capaz y están ampliando su alcance dentro de las arquitecturas de sistemas.

La IaC ofrece las siguientes ventajas frente a la configuración manual:

IaC Vs Op. Manual			
IaC		Op.Manual	
1	**Alta eficiencia:** automatiza la mayor parte de la administración de los recursos, que conlleva a optimizar el ciclo de vida del desarrollo del software.	1	**Costos altos:** la necesidad de capital humano puede ir hacia otras necesidades del negocio más importantes.
2	**Reutilización**: una vez se ha escrito el código para una infraestructura, este se puede ejecutar en cualquier momento y todas las veces que se desee.	2	**Inconsistencias:** debido al error humano, conducen a desviaciones del estándar de configuración.
3	**Control de versiones:** donde haya código, también es posible controlar las versiones.	3	**Falta de agilidad:** se limita limitar la velocidad a la que la organización puede lanzar nuevas versiones de servicios en respuesta a las necesidades e indicadores de mercado.
4	**Minimizar costes/esfuerzo**: automatizar la administración de la infraestructura ahorra muchos costes y horas de trabajo.	4	**Dificultad:** en alcanzar y mantener conformidad a la corporación o estándares de la industria debido a la falta de procesos repetibles.

Tabla 1. Enfoques de la IaC. Fuente: elaboración propia.

La infraestructura como código (en siglas, IaC) trata la infraestructura de configuración de sistemas como un software de programación. Esto genera una delgada línea entre los límites de la escritura de aplicaciones y la creación de entornos en los que se ejecutan. Se trata de una parte fundamental de la computación en la nube y esencial para DevOps. La IaC es el marco que ha dado origen a DevOps.

Figura 1. Enfoques de la IaC. Fuente: elaboración propia.

8.3. Ciclo de vida de los recursos de infraestructura

Las etapas del ciclo de vida de los recursos son las siguientes:

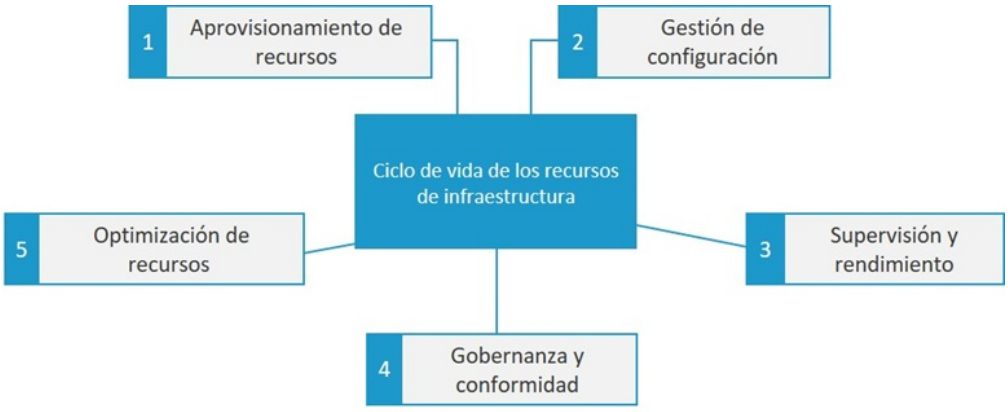


Figura 2. Ciclo de vida de los recursos de infraestructura. Fuente: elaboración propia.

Cada etapa involucra procedimientos que pueden aprovecharse en código, lo cual extiende los beneficios de la IaC de su rol tradicional de aprovisionar al ciclo de vida entero. Cada ciclo de vida se beneficia de la consistencia y repetitividad que la infraestructura como código ofrece.

- **Aprovisionamiento de recursos.** Etapa en la que los administradores utilizan este principio de la IaC para aprovisionar recursos de acuerdo con sus especificaciones y necesidades. AWS CloudFormation puede ser un claro ejemplo para aplicar.

- **Gestión de configuración.** Los recursos se vuelven componentes de un sistema de gestión de configuración que soportan actividades tales como optimización y actualización.

- **Supervisión y rendimiento.** Herramientas de supervisión y rendimiento validan el estado operacional de los recursos examinando ítems como métricas, transacciones sintéticas y archivos de registro.

Ideas clave

- **Gobierno y conformidad.** Los marcos de trabajo de conformidad y gobierno gestionan la validación adicional a fin de asegurar la relación y consonancia con la corporación y los estándares de la industria, así como requerimientos regulatorios.

- **Optimización de recursos.** Los administradores revisan datos de rendimiento e identifican cambios necesarios para optimizar el ambiente alrededor de los criterios tales como rendimiento y los costes.

8.4. Tipos de arquitecturas en Cloud

Con la aparición de la IaC, surgen diferentes tipos de arquitecturas en la nube, con conceptos y enfoques diferentes que solucionan el paradigma de cómo y con qué arquitectura empezar a trabajar. Nos encontramos con tres tipos de infraestructuras en la nube, cada una con un fin específico.

Infraestructura como Servicio (IaaS)

También conocida como HaaS (del inglés *Hardware as a Service*), esta arquitectura se basa en el modelo de dotar de forma externalizada a sus usuarios/empresas, del hardware necesario. IaaS proporciona hardware, almacenamiento, servidores y espacio de centro de datos o componentes de red. Como inconveniente podemos destacar que se requiere de los mismos conocimientos informáticos en sistemas operativos y redes informáticas que necesitábamos con una arquitectura tradicional.

Plataforma como Servicio (PaaS)

Abarca el ciclo completo de desarrollo de software para ser puesto en producción. El hardware y servidor donde se aloje es específico del proveedor, y por ello no causa ningún problema al software que alojemos y despleguemos en él. El concepto *serverless* es un PaaS para aplicaciones orientadas a Internet o web, en el que el coste es por llamada a la aplicación. Esto significa que, si no tiene llamadas, el servicio se apaga y queda en espera automáticamente.

Software como Servicio (SaaS)

Es un modelo de distribución de software donde la aplicación, el servidor y los datos son gestionados por un servidor externo. Se requiere de formación y adaptación para el uso correcto del producto y su pago, normalmente, es por usuario o licencia.

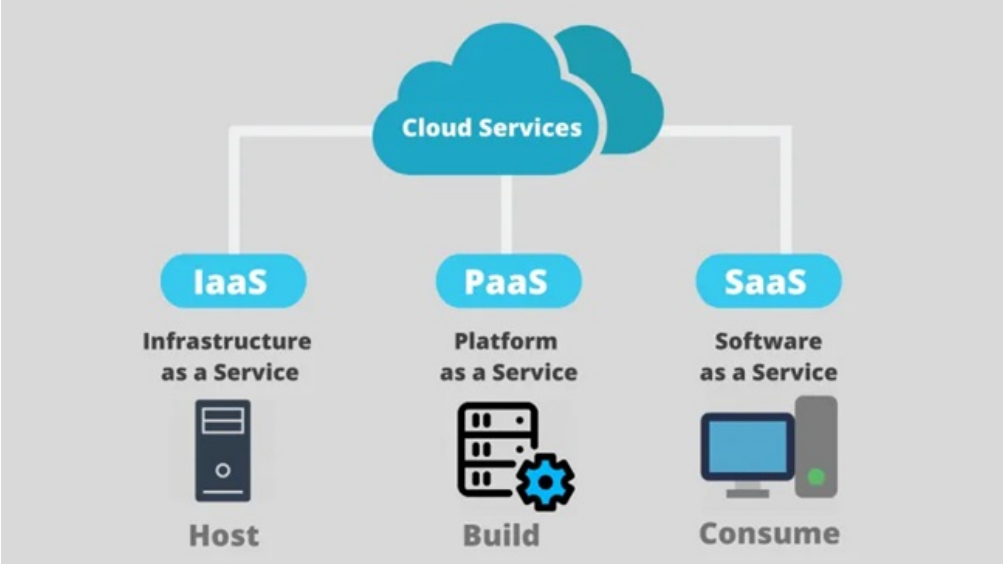


Figura 3. Overview arquitecturas Cloud. Fuente: Preesoft, s.f.

8.5. Herramientas para IaC

Independientemente de la herramienta que usemos, es necesario trabajar con algún lenguaje para la descripción de la configuración, pudiendo ser Json, Yaml o algún nuevo lenguaje propio del software. Dentro del archivo de configuración es donde definiremos todos los componentes necesarios para crear nuestra infraestructura. Las herramientas y *frameworks* especiales de IaC ofrecen sus propios lenguajes de configuración, que permiten administrar recursos de múltiples proveedores y eliminan la necesidad de conocer las API correspondientes en profundidad.

Figura 4. Mapa de herramientas IaC. Fuente: EdynTechnology, s. f.

Las herramientas más utilizadas actualmente son:

- **Azure Resource Manager:** servicio de Azure, plataforma en la nube de Microsoft, que proporciona la administración de infraestructuras mediante plantillas, de forma que implementa y supervisa todos los recursos. Esto da coherencia a la hora de reimplementar recursos ya existentes y permite definir las dependencias de estos.

Ideas clave

- **Google Cloud Deployment Manager:** Deployment Manager es, para la plataforma Google Cloud, lo que CloudFormation es para AWS. Con esta herramienta gratuita, los usuarios de recursos de IaaS de Google pueden administrarlos fácilmente mediante archivos de configuración central en el lenguaje de marcado Yaml.

- **Terraform:** la versión básica de Terraform, un software de código libre desarrollado por HashiCorp, está disponible de forma gratuita. Sus dos versiones de pago proporcionan funciones de configuración y organización, además de características para el trabajo grupal.

- **AWS CloudFormation:** CloudFormation es la herramienta interna de IaC de Amazon Web Services (en siglas, AWS) y, como tal, es prácticamente imprescindible para cualquiera que trabaje con productos de AWS como ELB, S3 o EFS. Utilizarla no conlleva ningún coste adicional, tan solo hay que pagar por los recursos reservados.

- **Heat:** implementa un motor de orquestación para poder lanzar múltiples aplicaciones en la nube basadas en plantillas, proporcionando una compatibilidad total con las plantillas de AWS CloudFormation. Proporciona, a su vez, una API nativa y una API compatible con AWS CloudFormation. Heat es el proyecto más ambicioso de OpenStack.

- **Chef Infra:** la solución de IaC de la empresa estadounidense Chef, está disponible desde abril de 2019 bajo la licencia gratuita Apache 2.0 y es utilizado por Facebook, entre otras empresas. Entre las plataformas compatibles se incluyen Google Cloud, Microsoft Azure, Amazon EC2 y OpenStack.

- **Puppet:** herramienta open source de gestión desarrollada en Ruby para la administración de sistemas de forma declarativa que, al estar basada en modelos, no requiere un alto conocimiento de programación para su uso.

Ideas clave

- **Red Hat Ansible Tower:** la herramienta de infraestructura como código de Ansible forma parte del catálogo de desarrollo de software Red Hat desde el año 2015. Ofrece un panel de control, su propia línea de comandos y una potentísima API Rest. En este caso, ambos paquetes disponibles, tanto el estándar como el extendido, son de pago.

8.6. IaC en AWS

Amazon Web Services (en adelante, AWS) proporciona la funcionalidad de IaC a través del servicio de CloudFormation. Este servicio contempla todos los pilares y fundamentos de IaC vistos anteriormente como:

- **Simplificación y rapidez.** Una aplicación web escalable que también incluye una base de datos de última generación, puede utilizar un grupo de escalado automático, un balanceador de carga elástico y una instancia de base de datos de Amazon Relational Database Service. Todas estas tareas pueden añadir complejidad y tiempo, incluso antes de conseguir que la aplicación funcione. En su lugar, podemos crear o modificar una plantilla de AWS Cloudformation existente. Una plantilla describe todos los recursos y propiedades. Cuando usamos esa plantilla para crear una pila de Cloudformation AWS, AWS Cloudformation provee el grupo de Auto Scaling, el balanceador de carga y la base de datos automáticamente. Después de que la pila se haya creado con éxito, los recursos AWS estarán en funcionamiento. Al utilizar AWS Cloudformation, podemos gestionar fácilmente una colección de recursos como una sola unidad.

- **Replicación y escalabilidad.** Podemos replicar nuestras aplicaciones en varias regiones de modo que, si una región no está disponible, sus usuarios todavía pueden utilizar la aplicación en otras regiones. Utilizando AWS Cloudformation, podemos reutilizar la plantilla para configurar los recursos de forma consistente y repetida. Una vez creados los recursos, podemos proveer los mismos recursos una y otra vez en múltiples regiones.

Ideas clave

- **Automatización mediante las API.** Una de las características que más flexibilidad proporciona en CloudFormation es la posibilidad de crear recursos personalizados. En el caso de que no sea posible crear un recurso con las opciones o características de forma nativa, mediante los tipos de recursos que proporciona CloudFormation, se pueden crear a través de un tipo de recurso «Custom». Para crear un recurso de este tipo se define primero en la plantilla un recurso Lambda (es donde se programan las llamadas a la API de AWS necesarias para la creación del recurso final), luego se referencia este recurso Lambda desde el recurso «Custom» y se le proporcionan los parámetros que necesite la función Lambda.

- **Actualizaciones controladas y *rollbacks*.** Se pueden actualizar los recursos subyacentes de forma incremental. Por ejemplo, podemos necesitar un cambio a un tipo de instancia de mayor rendimiento en la configuración de lanzamiento de Auto Scaling para que pueda reducir el número máximo de instancias en su grupo de Auto Scaling. Si se producen problemas después de completar la actualización, es posible que tengamos que hacer *rollbacks* de la infraestructura a la configuración original. Cuando utilizamos AWS CloudFormation, la plantilla de AWS CloudFormation describe exactamente qué recursos están provisionados y su configuración. Debido a que estas plantillas son archivos de texto, solo es necesario rastrear las diferencias entre las plantillas para ver los cambios en la infraestructura, similar a la forma en la que los desarrolladores controlan las revisiones al código fuente. Por ejemplo, puedes usar un sistema de control de versiones con plantillas para saber exactamente qué cambios se hicieron, quién los hizo y cuándo. Si en algún momento necesitáramos revertir cambios en la infraestructura, podemos usar una versión anterior de la plantilla.

AWS ClouFormation utiliza otros servicios de AWS complementarios, como ya hemos explicado en temas anteriores:

- AWS S3

- AWS CodeCommit

- AWS Lambda

AWS Cloudformation está compuesto principalmente por dos pilares:

Plantillas

Son los ficheros donde se describen las infraestructuras en sí mismas. Nombraremos algunas de las características fundamentales:

- Son susceptibles de ser anidadas e importadas por otras plantillas.
- El motor de plantillas ofrece funciones auxiliares similares a las de cualquier lenguaje de programación.
- Permiten definir parámetros y, a través de estos, se pueden escribir plantillas reutilizables para distintos entornos. Se pueden definir condiciones y utilizarlas para crear un recurso u otro en base a una condición.

Stacks

Es la implementación de una plantilla combinada con los parámetros necesarios. Una vez creada, permite comprobar y acceder a la información sobre: qué plantilla se ha usado, qué parámetros tenía, cuáles son los recursos creados y la información de salida que proporciona.

AWS CloudFormation es un servicio para modelar y configurar los recursos de AWS de forma sencilla, mediante la creación de una plantilla que describa todos los recursos de AWS que se necesiten (como las instancias de Amazon EC2 o DB de Amazon RDS), y AWS CloudFormation se encarga de aprovisionar y configurar esos recursos automáticamente.

Figura 5. Imagen de AWS CloudFormation. Fuente: Amazon Web Services, s. f.

Funcionamiento de AWS CloudFormation

Al crear una pila o *stack*, AWS CloudFormation realiza llamadas de servicio subyacentes a AWS para proporcionar y configurar sus recursos. AWS CloudFormation solo puede realizar acciones para las que tiene permiso, es decir, para crear instancias EC2 usando AWS CloudFormation, se necesitan permisos para realizar dicha acción. La administración de permisos se realiza a través de AWS Identity and Access Management (en siglas, IAM).

Las llamadas que hace AWS CloudFormation son declaradas en su totalidad en la plantilla. Por ejemplo, supongamos que tenemos una plantilla que describe una instancia EC2 con un tipo de instancia t1.micro. Cuando se utiliza esa plantilla para crear una pila, AWS CloudFormation llama a Amazon EC2 creando una API instancia y especifica el tipo de instancia como t1.micro. El siguiente diagrama resume el flujo de trabajo de AWS CloudFormation para crear pilas.

Figura 6. Flujo de trabajo en CloudFormation. Fuente: Amazon Web Services, s. f.

- Crea una plantilla de AWS CloudFormation (un documento con formato JSON o YAML) en AWS CloudFormation Designer o escribe una en un editor de texto. También podemos elegir una plantilla proporcionada por AWS CloudFormation. La plantilla describe los recursos que queremos y su configuración.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "A simple EC2 instance",
  "Resources" : {
    "MyEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : "ami-0ff8a91507f77f867",
        "InstanceType" : "t1.micro"
      }
    }
  }
}
```

Figura 7. Ejemplo en JSON usando plantillas para crear una instancia EC2. Fuente: Amazon Web Services, s.f.

- Guarda la plantilla localmente o en Amazon S3. Si es una creación desde cero, se guardará con cualquiera de las siguientes extensiones: .json, . yaml, o .txt.
- Crea una pila de AWS CloudFormation especificando la ubicación de la plantilla como ruta local o URL de Amazon S3. Si la plantilla contiene parámetros, es posible especificar valores de entrada en el momento de crear la pila. Los parámetros permiten pasar valores a la plantilla para personalizar los recursos cada vez que cree una pila. Adicionalmente, es posible crear pilas usando AWS CloudFormation, API o AWS CLI.

Una vez que los recursos hayan sido creados, AWS CloudFormation informa que la pila ha sido creada. Posteriormente, será posible empezar a utilizar los recursos de la pila creada. Si en el momento de la creación hubiese algún problema, AWS CloudFormation realizaría un *rollback* y eliminaría los recursos que se hayan creado.

Best practices en AWS CloudFormation

L a s *best practices* o buenas prácticas de AWS CloudFormation son recomendaciones para tener en cuenta a la hora utilizar dicha plataforma. Estas, sin dudas, fomentan el uso la plataforma de una forma más eficaz, eficiente y segura. Veamos cuáles son:

- Planificación y organización
  - Utilizar IAM para controlar el acceso.
  - Organizar las pilas por ciclo de vida y titularidad.
  - Utilizar referencias de pilas cruzadas para exportar recursos compartidos.
  - Reutilizar plantillas para replicar pilas en otros entornos.
  - Verificar las cuotas de los tipos de recursos.
  - Utilizar pilas anidadas para reutilizar patrones de plantillas comunes.

- Creación de plantillas
  - Utilizar AWS::CloudFormation::Init para implementar aplicaciones de software en las instancias de Amazon EC2.
  - Utilizar tipos de parámetros específicos y sus limitaciones de AWS.
  - No integrar credenciales en las plantillas.
  - Utilizar los scripts más recientes y validar las plantillas antes de usarlas.
- Gestión de pilas
  - Administrar todos los recursos de las pilas a través de AWS CloudFormation.
  - Utilizar las políticas de pilas de AWS.
  - Crear conjuntos de cambios antes de actualizar.
  - Actualizar con regularidad las instancias Linux de Amazon EC2.
  - Utilizar AWS CloudTrail para registrar las llamadas.

Ejemplo de acceso a AWS y la *template* de AWS CloudFormation

A continuación, veremos más a fondo un ejemplo de plantillas disponible dentro de AWS CloudFormation. Se debe tener en cuenta que, dependiendo de la región en la que estén nuestros servicios vinculados, tendremos unas plantillas genéricas y otras específicas de la región.

Para poder acceder a AWS CloudFormation, lo primero que debemos hacer es acceder a nuestra cuenta raíz de AWS y configurarla para crear un subdominio y un usuario del subdominio que será con el que se trabajará en adelante. Es recomendable que la cuenta *root* no se utilice para otro propósito que no sea el de la administración.

Para crear un subdominio y un usuario, debemos seguir los siguientes pasos:

- Accederemos a nuestra cuenta *root* y seleccionaremos el servicio IAM para configurar el subdominio y el usuario real con el que trabajaremos:

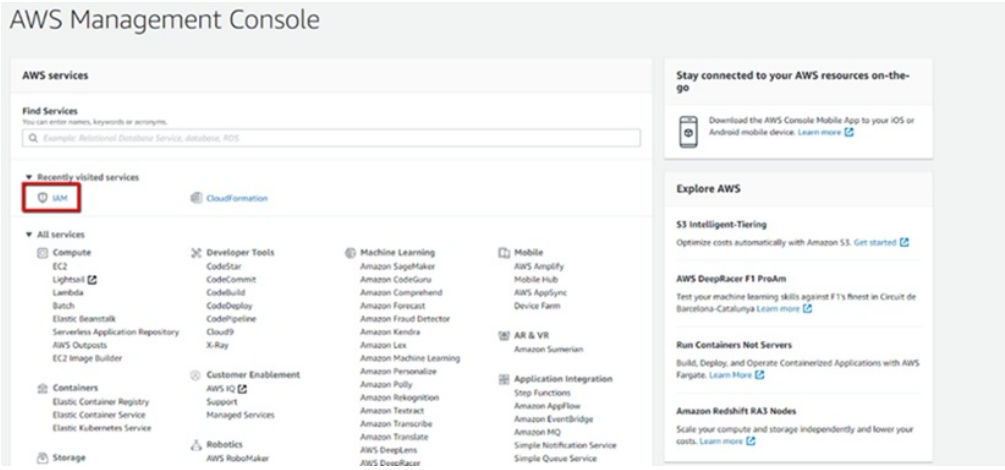


Figura 8. Consola de administración de AWS. Fuente: elaboración propia.

- Dentro del servicio IAM crearemos de un grupo de administración y un usuario; y configuraremos las políticas de seguridad y acceso. Una vez hecho esto, nuestra consola de IAM debe quedar de la siguiente forma:

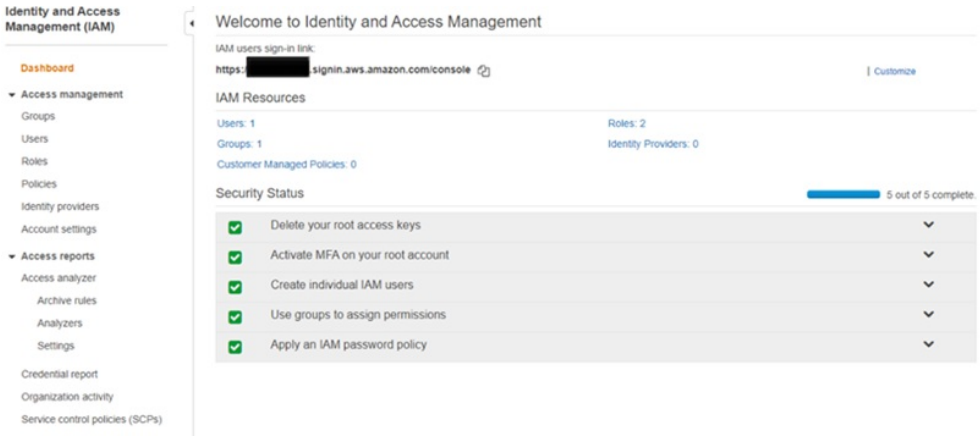


Figura 9. Consola de administración de IAM. Fuente: elaboración propia.

- Posteriormente, debemos copiar la URL que nos ha generado para acceder con nuestro nuevo usuario. Cerraremos la sesión root y accederemos a nuestra URL, que debe de ser así:

  - https://xxxxxxxxxx.signin.aws.amazon.com/console

- Ahora, accederemos a la siguiente página:

Figura 10. Acceso a consola AWS mediante usuario IAM. Fuente: elaboración propia.

- Desde ahora, siempre trabajaremos con nuestro usuario IAM por motivos de seguridad.

- Seleccionaremos dentro de la consola de administración el servicio AWS CloudFormation y accederemos a él:

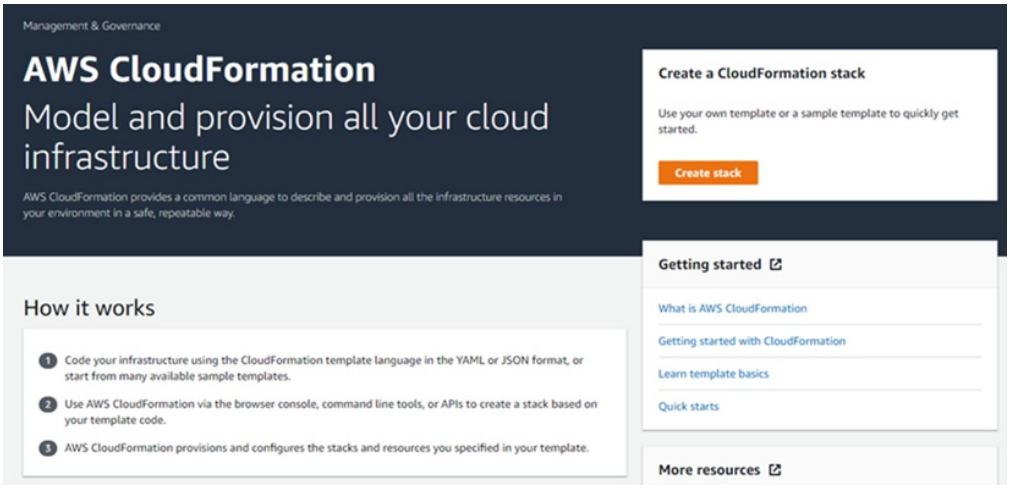


Figura 11. consola AWS CloudFormation. Fuente: elaboración propia.

Como ya se ha comentado anteriormente, las plantillas disponibles en AWS CloudFormation dependerán de la zona en la que nos encontremos, como se puede ver a continuación:

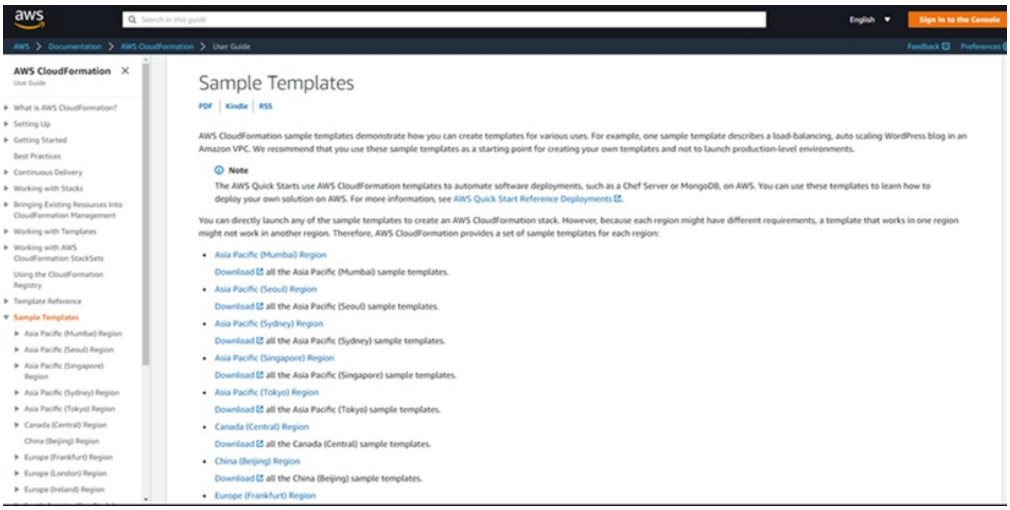


Figura 12. Plantillas ejemplo AWS CloudFormation por Zona Geográfica. Fuente: elaboración propia.

En nuestro caso, revisaremos una plantilla de la región de **Europa (Frankfurt).** AWS creará una instancia individual básica de Wordpress con una **base de datos MySQL** local para el almacenamiento y una **instancia Amazon EC2.** Tenemos la posibilidad de ver dicha plantilla en formato archivo, como se muestra en la siguiente imagen:

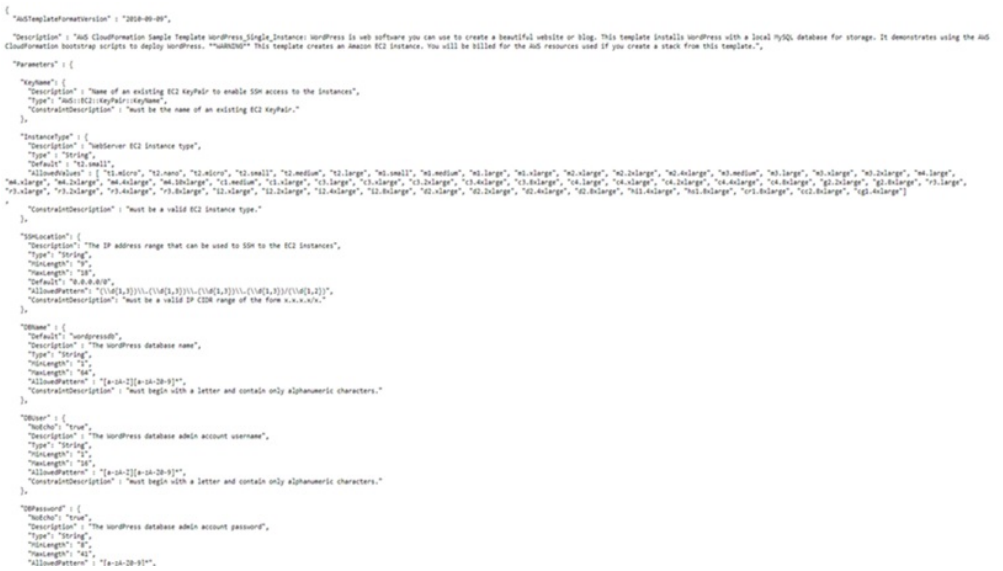


Figura 13. Fichero de configuración de una plantilla con AWS CloudFormation. Fuente: Amazonaws, s.f.

- Llegados a este punto, deberemos copiar la plantilla y configurarla según las necesidades o realizar la configuración completa mediante AWS CloudFormation Designer. Esta herramienta nos permitirá proceder con la configuración de una forma visual y sencilla mediante el *drag&drop* (arrastrar y soltar) de los elementos que queremos incorporar dentro de la interfaz.

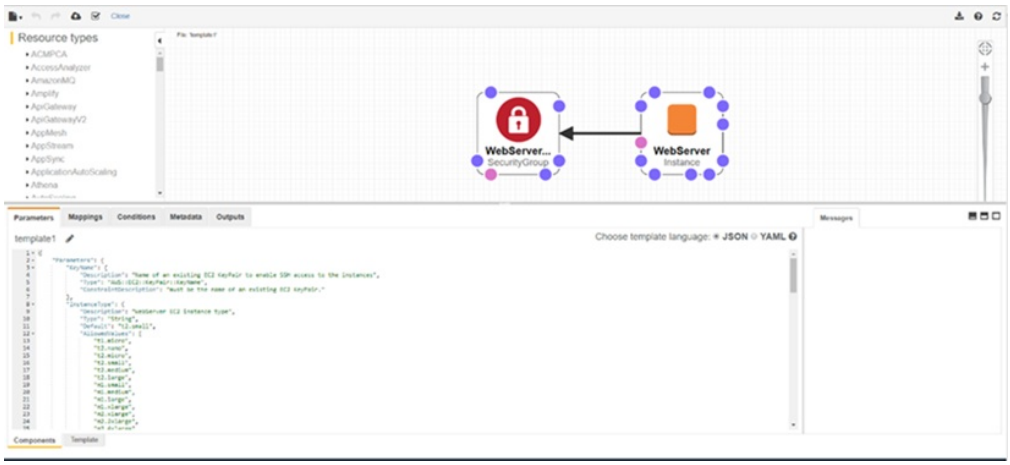


Figura 14. AWS CloudFormation Designer de AWS CloudFormation. Fuente: elaboración propia.

AWS CloudFormation Designer está compuesta por las siguientes áreas:

- **Diseño visual de la plantilla.** Aquí será posible realizar el diseño de nuestra solución a implementar. En el lado izquierdo de la página dispondremos de todos los recursos disponibles.

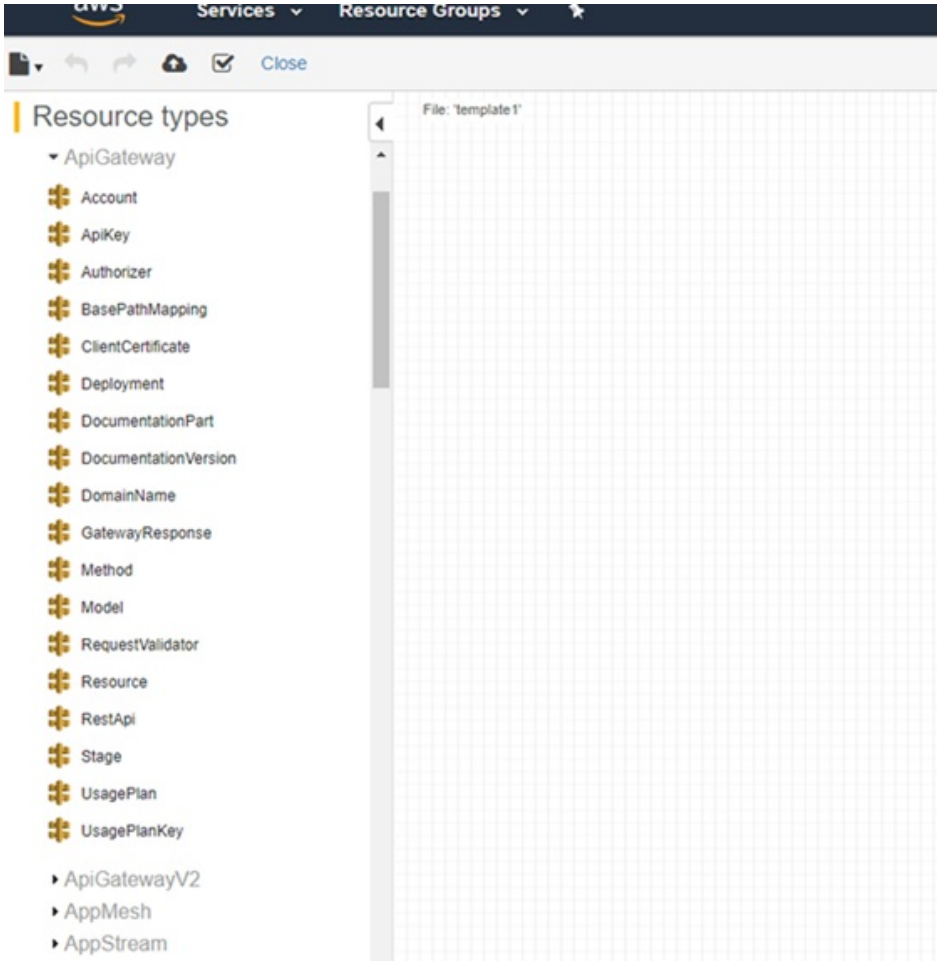


Figura 15. AWS CloudFormation Designer Selección de Recursos. Fuente: elaboración propia.

En la parte derecha de la interfaz haremos *drag&drop* para seleccionar los recursos que queremos desplegar, junto con la comunicación entre ellos y la configuración de seguridad, accesos, etc. La plantilla debería de quedar de esta forma:

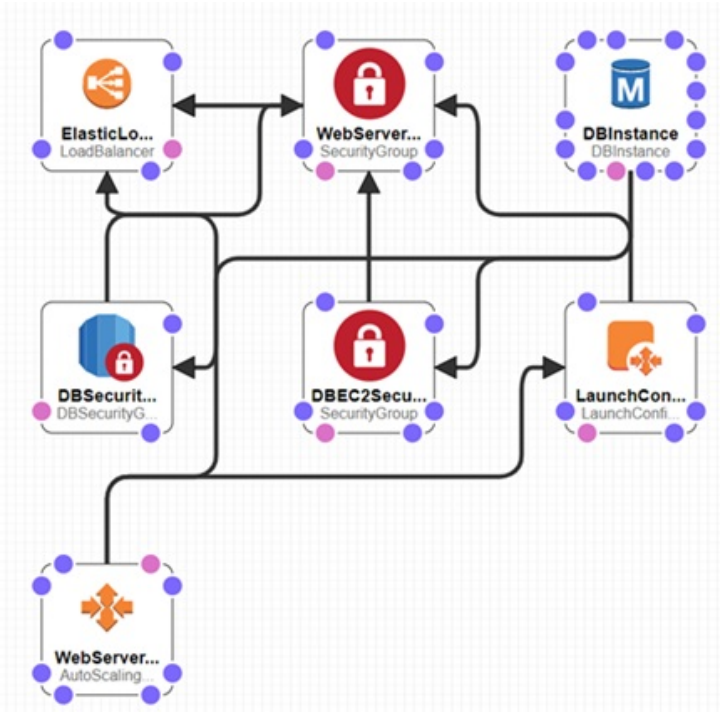


Figura 16. AWS CloudFormation Designer Visual Template. Fuente: elaboración propia.

- Configuración de parámetros, mapeos, condiciones, metadatos y salidas. Aquí es posible desarrollar los ficheros de configuración de forma manual, pudiendo elegir JSON o YAML como lenguaje.

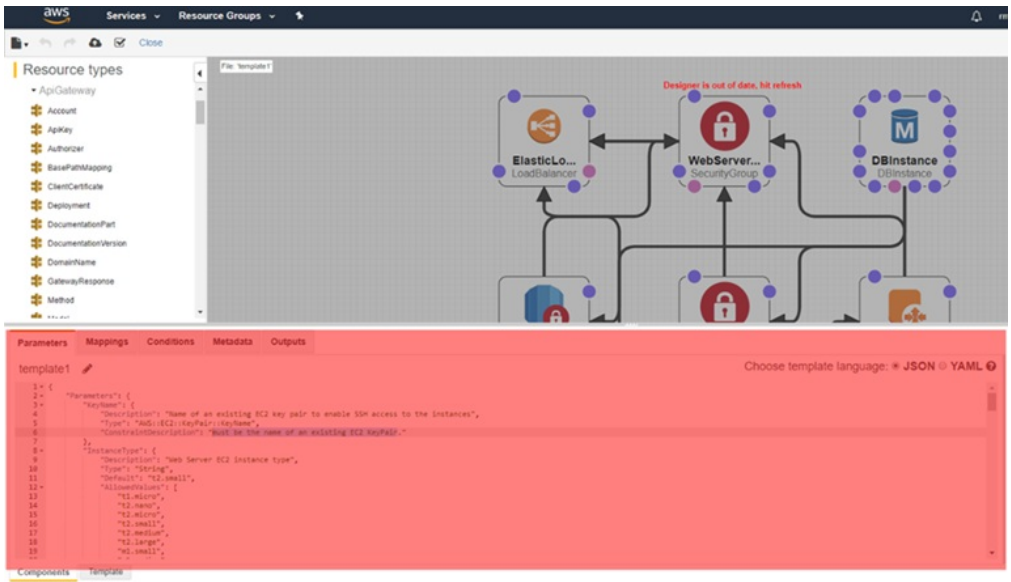


Figura 17. AWS CloudFormation Configuración total Template. Fuente: elaboración propia.

Una vez terminada la configuración, lo primero que debemos hacer es **comprobar si la plantilla es válida** antes de realizar el despliegue. Para ello, debemos marcar la casilla como en la imagen que aparece debajo:

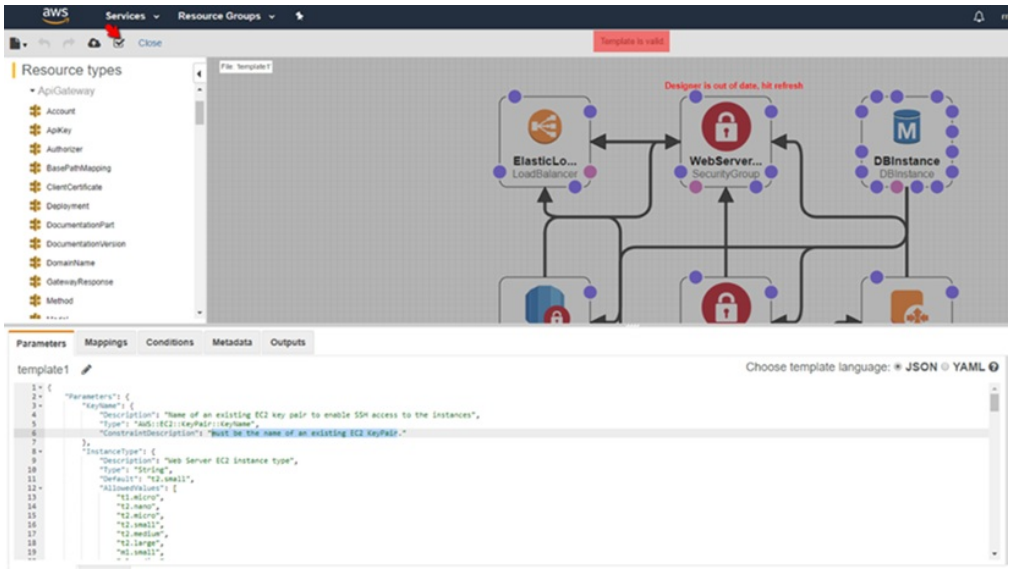


Figura 18. AWS CloudFormation validación Template. Fuente: elaboración propia.

El último paso que queda es crear el *stack* o pila y el tiempo de creación variará, dependiendo de los servicios que vayamos a desplegar y la complejidad de la configuración.

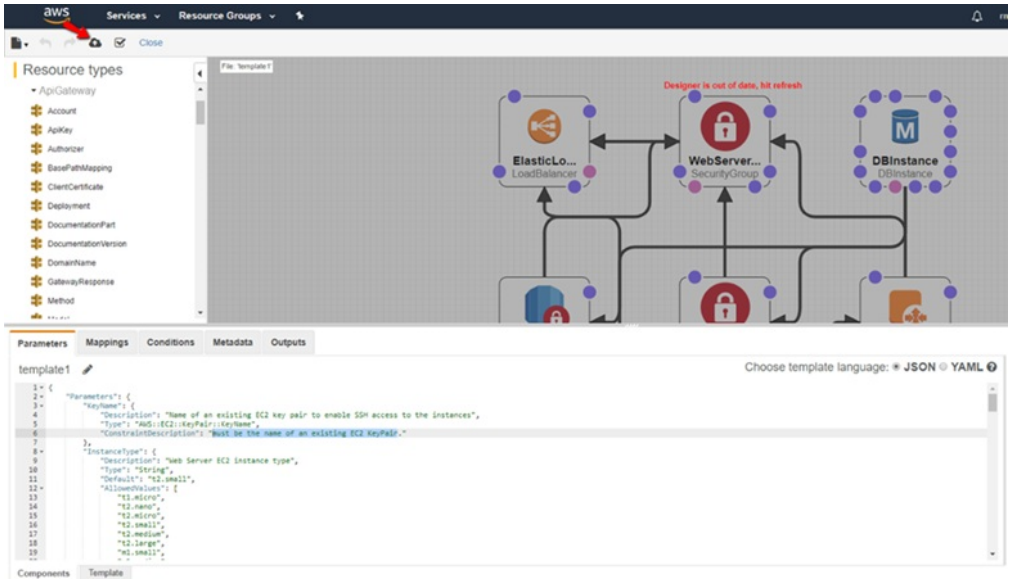


Figura 19. AWS CloudFormation creación Stack. Fuente: elaboración propia.

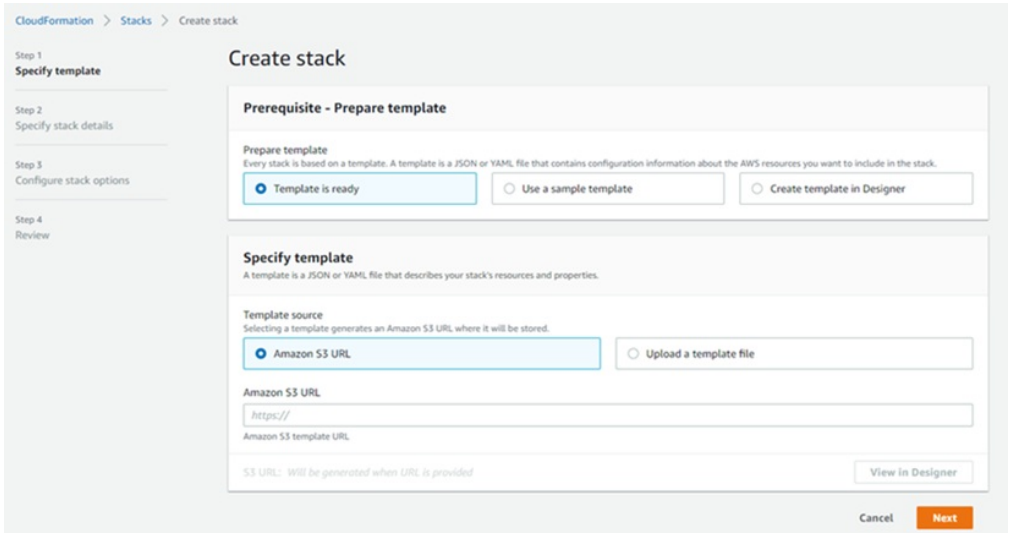
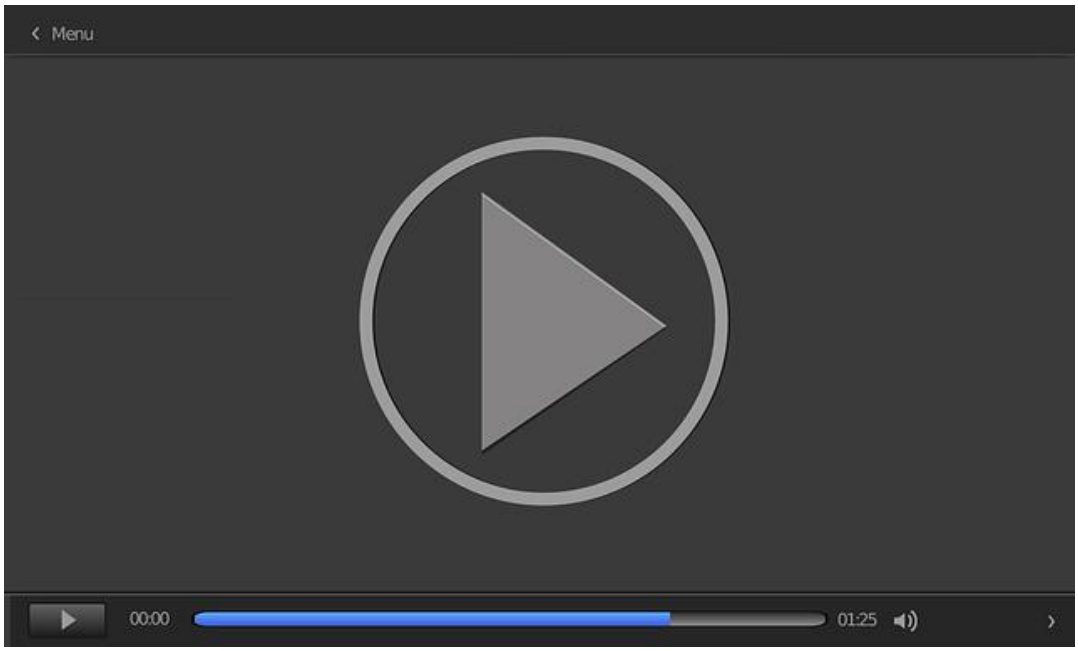


Figura 20. AWS CloudFormation creación Stack y carga de Template. Fuente: elaboración propia.

Ideas clave

En la píldora *DevOps en Spotify* analizaremos cómo Spotify (plataforma de renombre mundial dedicada a la comercialización de servicios audiovisuales) implementa la filosofía DevOps y cuáles son las claves de su éxito.



Accede al vídeo:https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=d3124a5d-5f6c-423b-8c6b-ad78013d2af6

8.7. Referencias bibliográficas

Edyn Technology. (s.f.). *Infraestructura como código (IaC)*. http://edyntechnology.com/servicios/infrastructure-as-code-iac/

Amazon Web Services. (s.f.). *¿Cómo funciona AWS CloudFormation?* https://docs.aws.amazon.com/es_es/AWSCloudFormation/latest/UserGuide/cfn-whatis-howdoesitwork.html

Amazon Web Services. (s.f.). *AWS CloudFormation*. https://aws.amazon.com/es/cloudformation/

Amazon Web Services. (s.f.). *AWS Template Format Version* . https://s3.eu-central-1.amazonaws.com/cloudformation-templates-eu-central-1/WordPress_Single_Instance.template

Andreessen, M. (2011). Why Software Is Eating the World. *The Wall Street Journal*. https://www.wsj.com/articles/SB10001424053111903480904576512250915629460

Definición de IaC

Andreessen, M. (2011). Why Software Is Eating the World. *The Wall Street Journal*. https://www.wsj.com/articles/SB10001424053111903480904576512250915629460

A través de la lectura de este artículo, puedes profundizar en la infraestructura como código.

Test

1. ¿Cuál es el objetivo de la Infraestructura como Código o IaC?

   A. Eliminar la necesidad configuración manual del hardware.

   B. La evolución natural de los sistemas.

   C. No tiene un objetivo claro.

   D. Ahorrar puestos de trabajo, ya que reduce la intervención humana.

2. ¿Es la IaC un software de programación?

   A. Sí, porque es con la IaC con la que desarrollamos en la nube.

   B. No, se trata de una parte fundamental de la computación en la nube y esencial para DevOps.

   C. Sí, porque la programación es lo único que puedo hacer.

   D. Sí, en todos los casos.

3. ¿Cuáles de estas opciones son ventajas en de la IaC?

   A. Optimización.

   B. Inconsistencia de recursos.

   C. Bajo coste y esfuerzo.

   D. Alta dificultad.

4. ¿Cuáles de estas herramientas son de IaC?

   A. Microsoft Azure.

   B. Terraform.

   C. Chef Infra.

   D. Json.

5. ¿Es la IaC un enfoque declarativo?

A. Sí, porque se centra en la declaración del destino final y AWS CloudFormation se basa en este enfoque.

B. No, es imperativo porque se centra en cómo la infraestructura se va a cambiar para cumplir este enfoque y, además, Azure se basa en este enfoque.

C. No, es inteligente porque es esencialmente el «qué» frente al «cómo» y frente al «por qué».

D. Todas las anteriores.

6. Relaciona Best Practice de AWS Cloudformation con su detalle:

Planificación y organización	1
Creación de plantillas	2
Organizar las pilas por ciclo de vida y titularidad	3

A	Administrar todos los recursos de las pilas, a través de AWS CloudFormation
B	Utilizar los scripts más recientes y validar las plantillas antes de usarlas.
C	Organizar las pilas por ciclo de vida y titularidad.

7. ¿Cuáles son las partes principales de AWS CloudFormation?

A. AWS Identity y AWS S3.

B. Plantillas y Stacks.

C. AWS CodeCommit.

D. Lambda.

8. ¿Qué pilares y fundamentos de IaC contempla AWS CloudFormation?

A. Replicación y escalabilidad.

B. Automatización mediante las API.

C. Actualizaciones controladas y *rollbacks*.

D. Las tres anteriores son ciertas.

9. ¿En qué entorno podemos crear una plantilla?

A. En JSON o YAML

B. En AWS CloudFormation Designer.

Es el entorno proporcionado por AWS.

C. En el Notepad.

D. No se pueden crear, solo se pueden utilizar las proporcionadas por AWS CloudFormation.

10. ¿Cuál es el flujo de trabajo de AWS CloudFormation?

A. Crear permisos, crear una cesta en S3 y crear una pila.

B. Crear una pila, guardarla y crear la plantilla.

C. Crear o utilizar una plantilla, guardarla en local (o en S3) y crear la pila basada en la plantilla.

D. No se puede hacer flujos de trabajo en AWS.