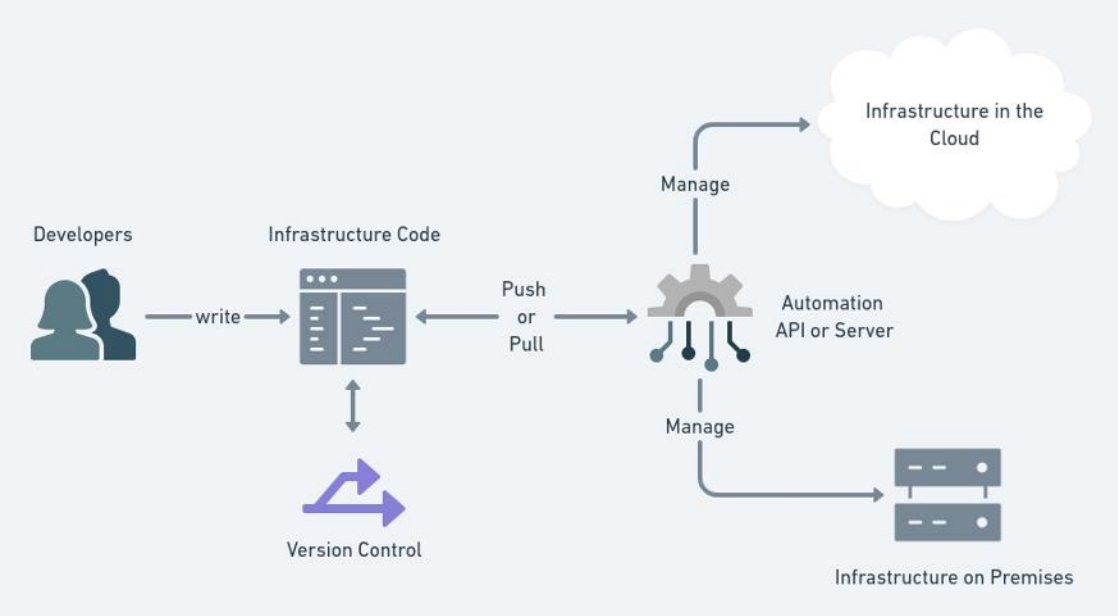
Infraestructura como Software



SOFTWARE ARCHITECTURE

- Sergio Castillo García UO276436
- Laís García Suárez UO276208
- Tsegan Manuel Antón De La Calle UO276213

Que es, Como surgió y Ventajas

La infraestructura definida por software, como su propio nombre indica, va de tratar a la configuración de los sistemas como si fuera código programable (véase Turing), aportando una serie de ventajas. La más importante es el poder probar acciones que de otra manera se tendrían que realizar de manera manual, lo que nos da un mayor grado de productividad y mejora en la calidad del servicio.

Está claro que hoy en día tratamos de automatizar toda acción que tengamos que repetir unas cuantas veces, así que este campo no iba a ser menos, y es que el hecho de tener que configurar y testear un sistema de forma manual es tanto arcaico como ineficiente. Cabe desatacar que esto no es automatizar la infraestructura, eso sería repetir acciones repetidamente, la infraestructura como código nos permite garantizar que estas acciones sean probadas antes de ser aplicadas.

Actualmente las empresas requieren de la virtualización tanto para infraestructuras internas como para servidores, por lo que es casi impensable que tengamos que crear y configurar estas máquinas a mano. Podemos utilizar tanto scripts como herramientas para esta tarea, algunas de los más famosas son Vagrant y Docker (imagina lo que tardarías en hacer lo mismo que hacen estos programas en pocos segundos)

Otra ventaja es el poder utilizar el código como documentación, lo que facilita el desarrollo de esta, así como la creación de diagramas, esquemas… Además del control de versiones (véase más adelante)

IaC y DevOps

Infrastructure as Code es una técnica muy habitual en la cultura DevOps.

DevOps es un modo de trabajo que consiste en aumentar la comunicación y colaboración de los desarrolladores y de los IT mediante técnicas DevOps y herramientas DevOps. Infrastructure as Code encaja perfectamente en este contexto ya que facilita a los IT involucrarse más temprano en el proceso de desarrollo y a los programadores tener un papel más activo en la configuración.

Turing Completo

Las herramientas IaC son en su mayoría Turing completo, es decir, son capaces de imitar una máquina de Turing universal. Esto facilita la integración de con sistemas de pruebas.

Tipos de IaC

Los lenguajes de herramientas IaC pueden ser declarativos o imperativos. Los declarativos consisten es definir el estado deseado y el sistema ejecuta los pasos necesarios para ello y en el caso de los imperativos, se definen los comandos para obtener el resultado deseado.

Las herramientas IaC se pueden dividir según su funcionamiento en *pull* o *push*. En las herramientas *pull*, los servidores creados obtienen sus datos de configuración de un servidor primario y en las herramientas *push*, es el servidor primario el que envía la configuración a la máquina.

Tool	Released by	Method	Approach	Written in
Chef	Chef (2009)	Pull	Declarative and imperative	Ruby
Otter	Inedo	Push	Declarative and imperative	-
Puppet	Puppet (2005)	Pull	Declarative and imperative	C++ & Clojure since 4.0, Ruby
SaltStack	SaltStack	Push and Pull	Declarative and imperative	Python
CFEngine	Northern.tech	Pull	Declarative	C
Terraform	HashiCorp (2014)	Push	Declarative	Go
Ansible / Ansible Tower	Red Hat (2012)	Push	Declarative and imperative	Python

Control de versiones

Al estar la creación de la infraestructura automatizada mediante scripts, compuestos por código, nos permite utilizar sistemas de control de versiones (git, cvs, svn etc) como si de software convencional se tratara.

Esto nos permite volver a una versión estable anterior de nuestra infraestructura en caso de detectar inconsistencias, en caso de necesitarlo tener disponibles para desplegar distintas versiones de sistema y mantener un histórico de la evolución de nuestro sistema.

Problemas/retos

Uno de los principales problemas es la aún limitada compatibilidad con sistemas de Microsoft (Docker está directamente en Linux, pero en Windows depende de Hyper-V) y aun cuando es posible aplicar estas técnicas en estos sistemas acaba resultando más incómodo y caro que su alternativa Linux.

Dejando a un lado el operativo en el que se ejecute, los propios servidores donde pretendas usar las herramientas de *infrastructure as code* pueden necesitar de preparación como instalar programas de virtualización u obtener permiso para desplegar la infraestructura virtualizada. Esto puede implicar unos razonablemente simples trámites o la limitación e incluso la imposibilidad de hacer las cosas como estaba planeado.

Otra consecuencia para considerar es el aprendizaje que requiere dominar estas herramientas, más aún si se espera integrar con sistemas PaaS que hace aumentar la complejidad del trabajo, el cual requiere una importante inversión de tiempo y dinero resultando en la no adopción por parte de muchas empresas.

Bibliografía

Nivenly.com - Kris Nóva — Linux, Kubernetes, Cyber security, Infrastructure, Hacking

Infrastructure as Code - Automation Is Not Enough (http_kief.com)

Meet Infrastructure as Code (https_devops.com)

TestPyramid (martinfowler.com)

Infrastructure as code - Wikipedia

Turing completo - Wikipedia, la enciclopedia libre

Top 10 Infrastructure as Code Tools to Boost Your Productivity (nexastack.com)

Qué es DevOps (y sobre todo qué no es DevOps) - Paradigma (paradigmadigital.com)