Infraestructura como Código IaC

Alexander Zelada

HashiCorp Terraform

2023

Mi CV

2022-Actualidad Devops Lead - TMC (España)

2020-2022 Ingeniero de Operaciones Cloud - Santander (España)

2019 Líder de Nube Pública - Canvia (Perú)

2017-2019 Ingeniero de Infraestructura - Everis- (Perú)

https://www.linkedin.com/in/lzeladam/

Lanscape

HashiCorp Terraform

GitLab

kubernetes

Del Hierro a la Nube

Iron Age	Cloud Age
Hardware Físico	Recursos Virtualizados
El aprovisionamiento toma semanas	El aprovisionamiento toma minutos
Procesos manuales	Procesos automatizados

Tabla 1. Cambios de tecnológicos en “The Cloud Age”

Fuente: Infrastructure as Code Dynamic Systems for the Cloud Age por Kief Morris

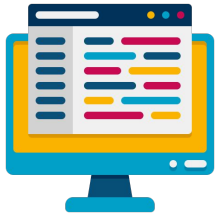
Formas de trabajo en “The Cloud Age”

Iron Age	Cloud Age
El costo del cambio es alto	El costo del cambio es bajo
Los cambios representan fracaso (los cambios deben ser 'gestionados' y 'controlados')	Los cambios representan aprendizaje y mejora.
Reducir las oportunidades de fracasar	Maximiza la velocidad de mejora
Entregas en grandes lotes, pruebas al final	Entregas en pequeños lotes, pruebas continuas
Ciclos de release largos	Ciclos de release cortos
Arquitecturas monolíticas (menos piezas móviles y más grandes)	Arquitecturas de microservicios (más partes, más pequeñas)
Configuración mediante GUI o física	Configuración como Código

Fuente: Infrastructure as Code Dynamic Systems for the Cloud Age por Kief Morris

¿Qué es IaC?

- Enfoque para la automatización de la infraestructura que se basa en prácticas de desarrollo de software.
- Utiliza código para describir la configuración y el aprovisionamiento de los sistemas de infraestructura.
- Permite una gestión más eficiente, escalable y consistente.



Beneficios de IaC

Entrega rápida de valor usando infraestructura IT

Reducción del esfuerzo y riesgo al hacer cambios

Recursos disponibles para usuarios de infraestructura

Herramientas comunes para desarrollo, operaciones y otros interesados

Sistemas confiables, seguros y rentables

Visibilidad de gobernanza, seguridad y cumplimiento

Mejora de velocidad para solucionar problemas y resolver fallos

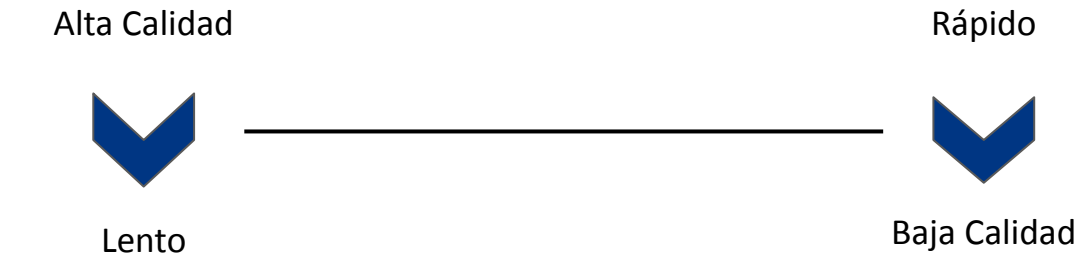
Objeciones comunes a la automatización

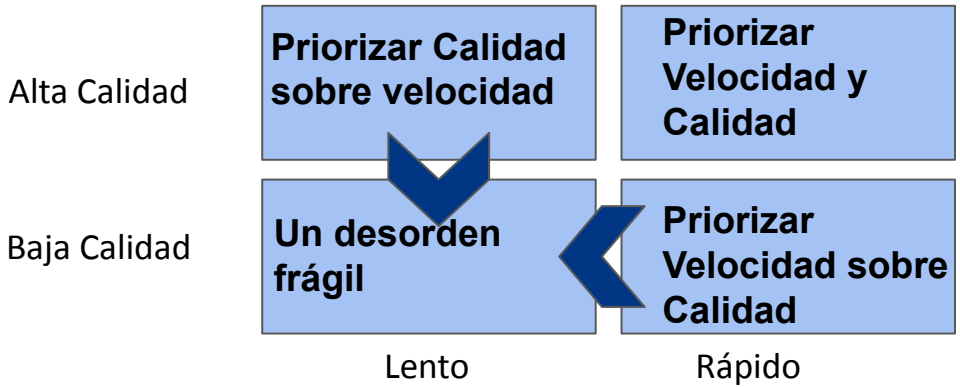
No hacemos cambios con la suficiente frecuencia como para justificar la automatización

Deberíamos construir primero y automatizar después

Debemos elegir entre velocidad y calidad.

Debemos elegir entre velocidad y calidad





Principios de la IaC

Define todo como código

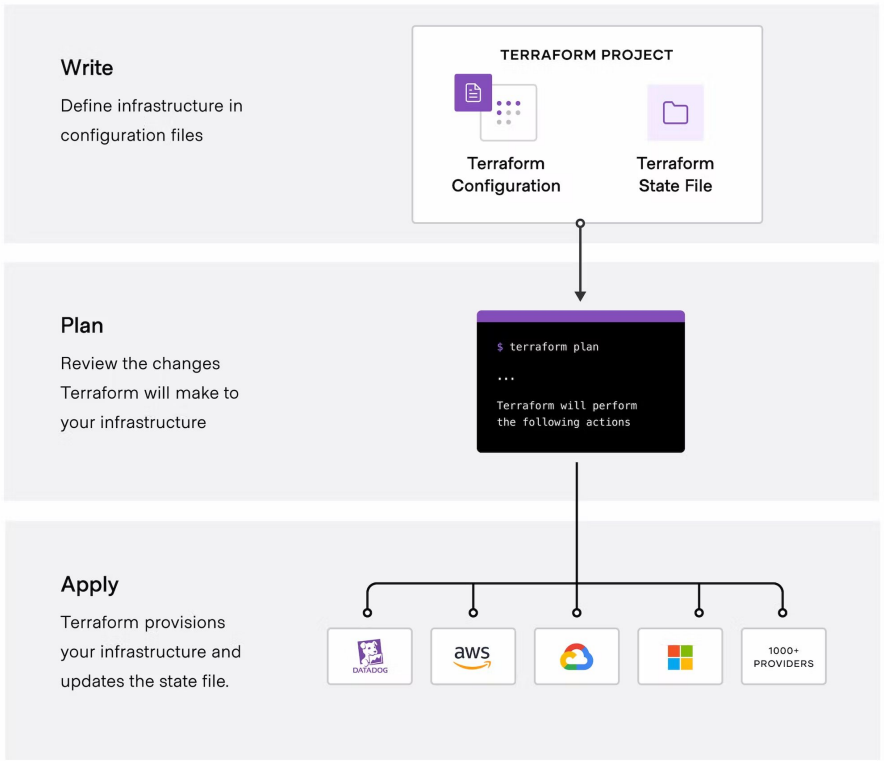
Prueba y entrega continuamente todo el trabajo en progreso

Construye piezas pequeñas y simples que puedas cambiar de manera independiente

¿Qué es Terraform?

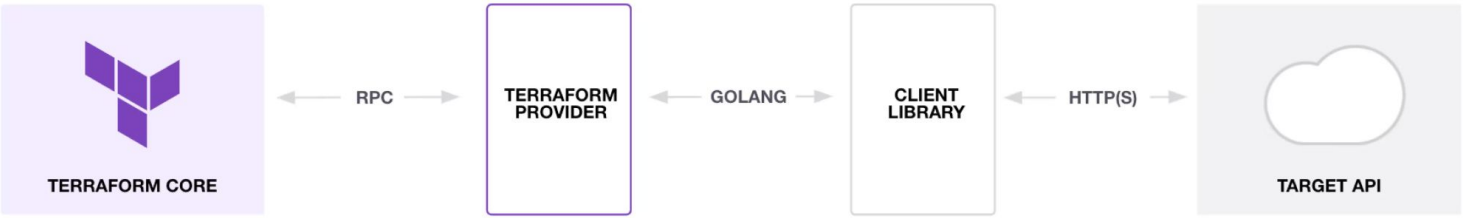
- Terraform es una herramienta de código abierto desarrollada por HashiCorp.
- Con Terraform, puedes definir, crear y gestionar la infraestructura como código
- Terraform es compatible con múltiples plataformas y proveedores de nube.
- El lenguaje declarativo utilizado por Terraform es simple y legible para los humanos.

Workflow de Terraform



¿Cómo funciona?

- Terraform está escrito en Go
- El binario de Terraform es un archivo ejecutable para crear, modificar y eliminar recursos de infraestructura.
- Analiza los archivos de configuración de Terraform
- Compara el estado actual de la infraestructura con la definición declarada en los archivos de configuración de Terraform.
- Determina la serie de cambios necesarios para hacer que el estado de la infraestructura coincida con la definición declarada.
- Envía las solicitudes correspondientes a la API del proveedor de nube para crear, modificar o eliminar los recursos de infraestructura.
- Actualiza el estado interno de Terraform para reflejar el nuevo estado de la infraestructura.
- Genera un plan que muestra los cambios propuestos antes de aplicarlos
- Finalmente, aplica los cambios de infraestructura necesarios para hacer que el estado de la infraestructura coincida con la definición declarada



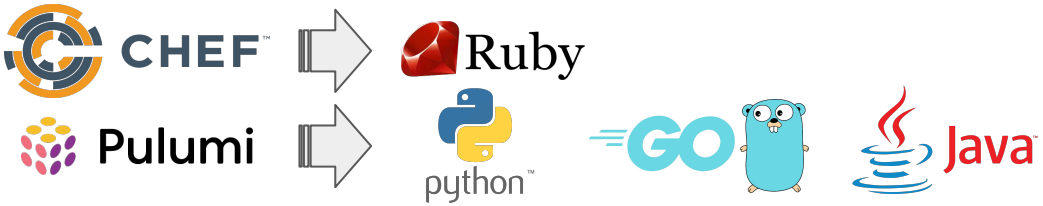
Procedural vs Declarativo

- En el estilo procedural se escribe código que indica paso a paso cómo realizar una tarea para lograr un estado deseado.

- En el estilo declarativo se escribe código que indica el estado final deseado y la herramienta de infraestructura como código se encarga de determinar cómo lograr ese estado.

GLP vs DSL

- General-Purpose Programming Language (GLP)



- Domain-Specific Language (DSL)

Una comparación de las formas más comunes de usar las herramientas de infraestructura como código más populares.

	Chef	Puppet	Ansible	Pulumi	CloudFormation	Heat	Terraform
Source	Open	Open	Open	Open	Closed	Open	Open
Cloud	All	All	All	All	AWS	All	All
Type	Config mgmt	Config mgmt	Config mgmt	Provisioning	Provisioning	Provisioning	Provisioning
Infra	Mutable	Mutable	Mutable	Immutable	Immutable	Immutable	Immutable
Paradigm	Procedural	Declarative	Procedural	Declarative	Declarative	Declarative	Declarative
Language	GPL	DSL	DSL	GPL	DSL	DSL	DSL

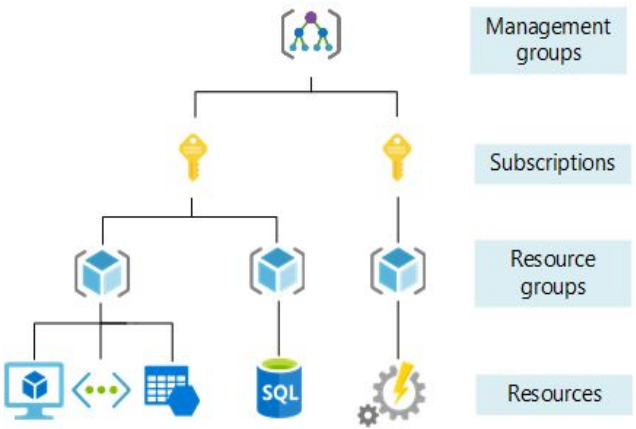
Fuente: Terraform: Up and Running por Yevgeniy Brikman

Instalar Terraform

DEMO

Azure

Niveles y Jerarquías de gestión



Recursos principales:

- Organize your Azure resources effectively - Cloud Adoption Framework
- Estado de Azure
- Directorio de Azure Cloud Services
- Calculadora de precios | Microsoft Azure
- Browse Azure Architectures - Azure Architecture Center | Microsoft Learn

Entorno Azure

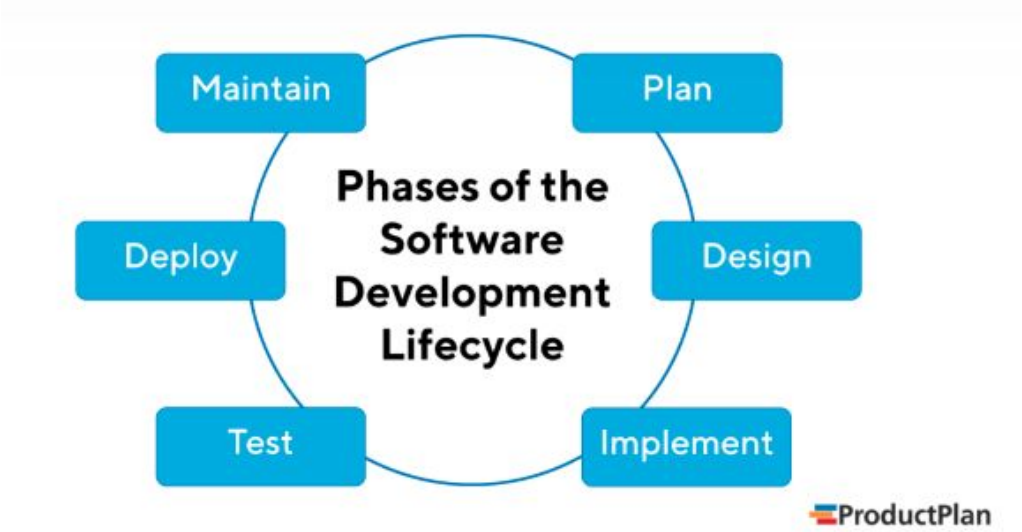
DEMO

DevOps

En la actualidad, muchas empresas están migrando de la gestión de sus propios data centers a la nube, AWS, Azure y GCP. Los equipos de operaciones están invirtiendo su tiempo en herramientas de software como Chef, Puppet, Terraform, Docker y Kubernetes en lugar de en hardware. Esto ha llevado a una disminución de la diferencia entre los equipos de desarrollo y operaciones, y la necesidad de trabajar juntos de manera más estrecha. Esto ha dado lugar al movimiento DevOps, donde ambos equipos pasan la mayor parte de su tiempo trabajando en software y colaborando estrechamente.

El objetivo de DevOps es hacer que la entrega de software sea mucho más eficiente.

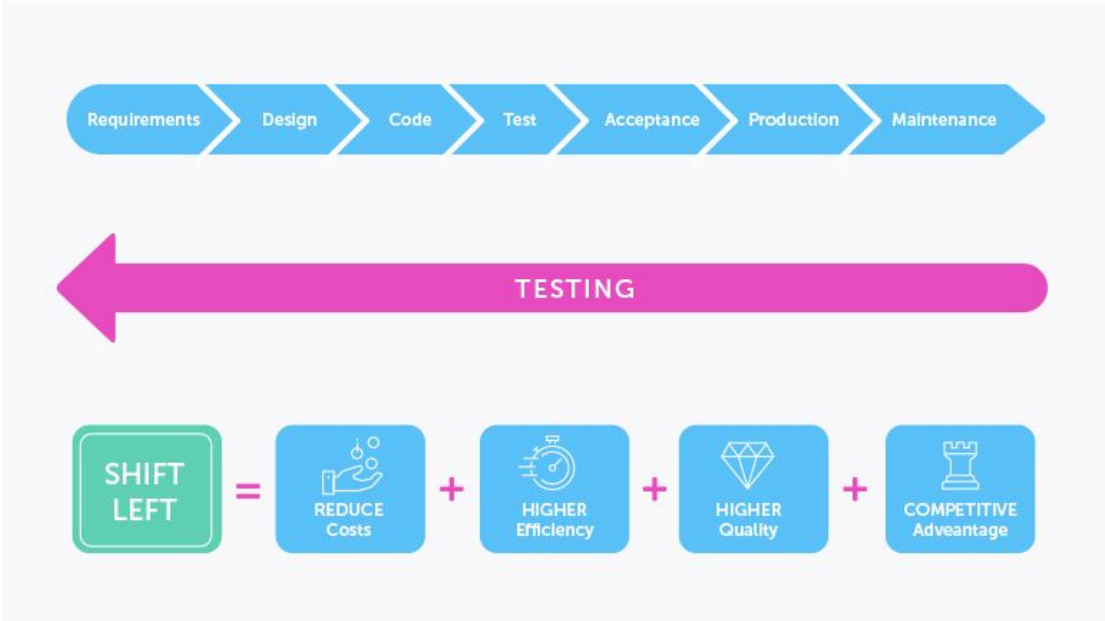
Ciclo de vida de software



En 1970 Winston W. Royce presenta el modelo de ciclo de vida de desarrollo de software.

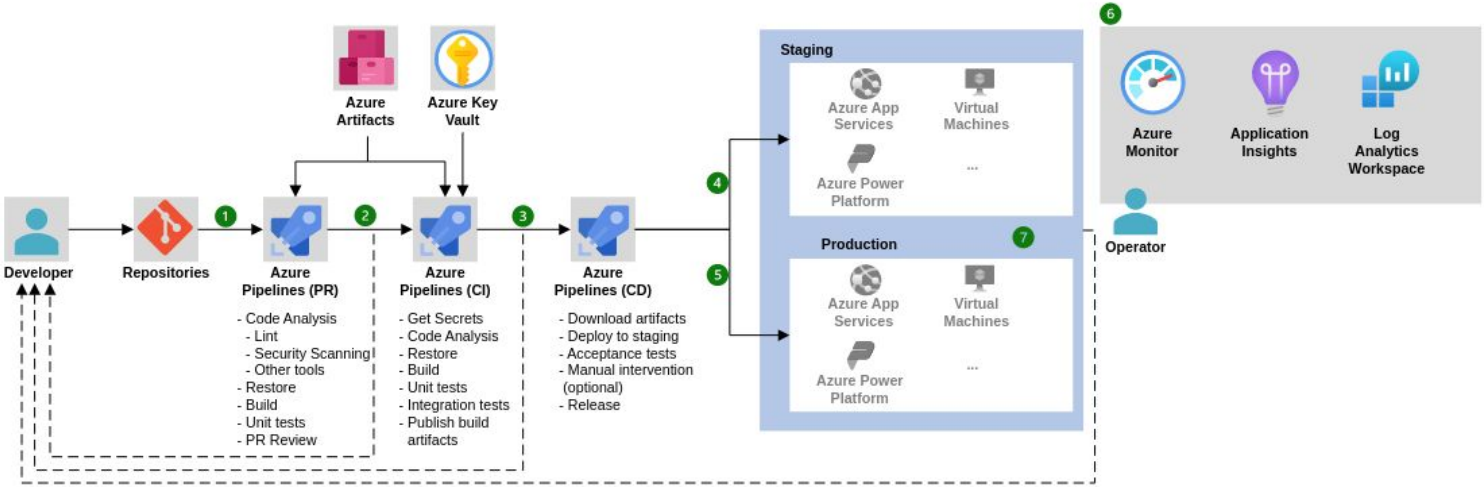
Fuente:
What is the Software Development Lifecycle? https://cutt.ly/Z84YZXE
Managing the development of large software systems https://cutt.ly/F84UaS6

Shift-left testing



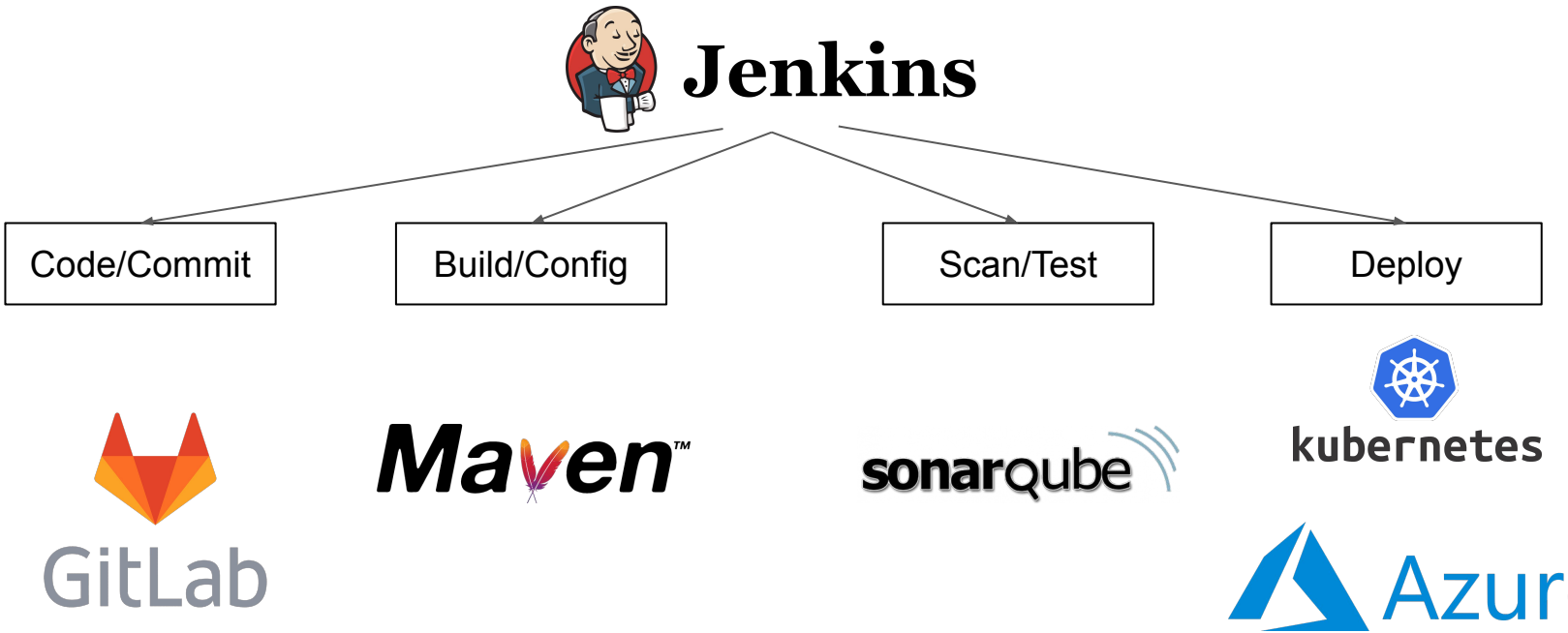
Fuente:
Why Does DevOps Recommend Shift-Left Testing Principles? https://cutt.ly/W84SJud

Arquitectura CI/CD con Azure Pipelines



Fuente:
CI/CD baseline architecture with Azure Pipelines http://bit.ly/3T9vF2g

Arquitectura CI/CD con Jenkins



Fuente:
CI/CD baseline architecture with Azure Pipelines http://bit.ly/3T9vF2g

Preparación del entorno

DEMO