

infraestructura como código

IaaS

ocultar/mostrar

- adaptar metodologías de desarrollo al mundo de sistemas
- control de versiones
- lenguajes de programación
- gestión de la configuración
- despliegues automatizados
- basado en pruebas: integración continua, entrega continua y despliegue continuo
- AUTOMATIZACIÓN

DevOps

ocultar/mostrar

desarrollo del software

- análisis
- diseño
- desarrollo
- pruebas
- despliegue

Metodologías ágiles

- Manifiesto
- XP
- Lean
- Scrum
- Test Driven Development

Integración continua (CI)

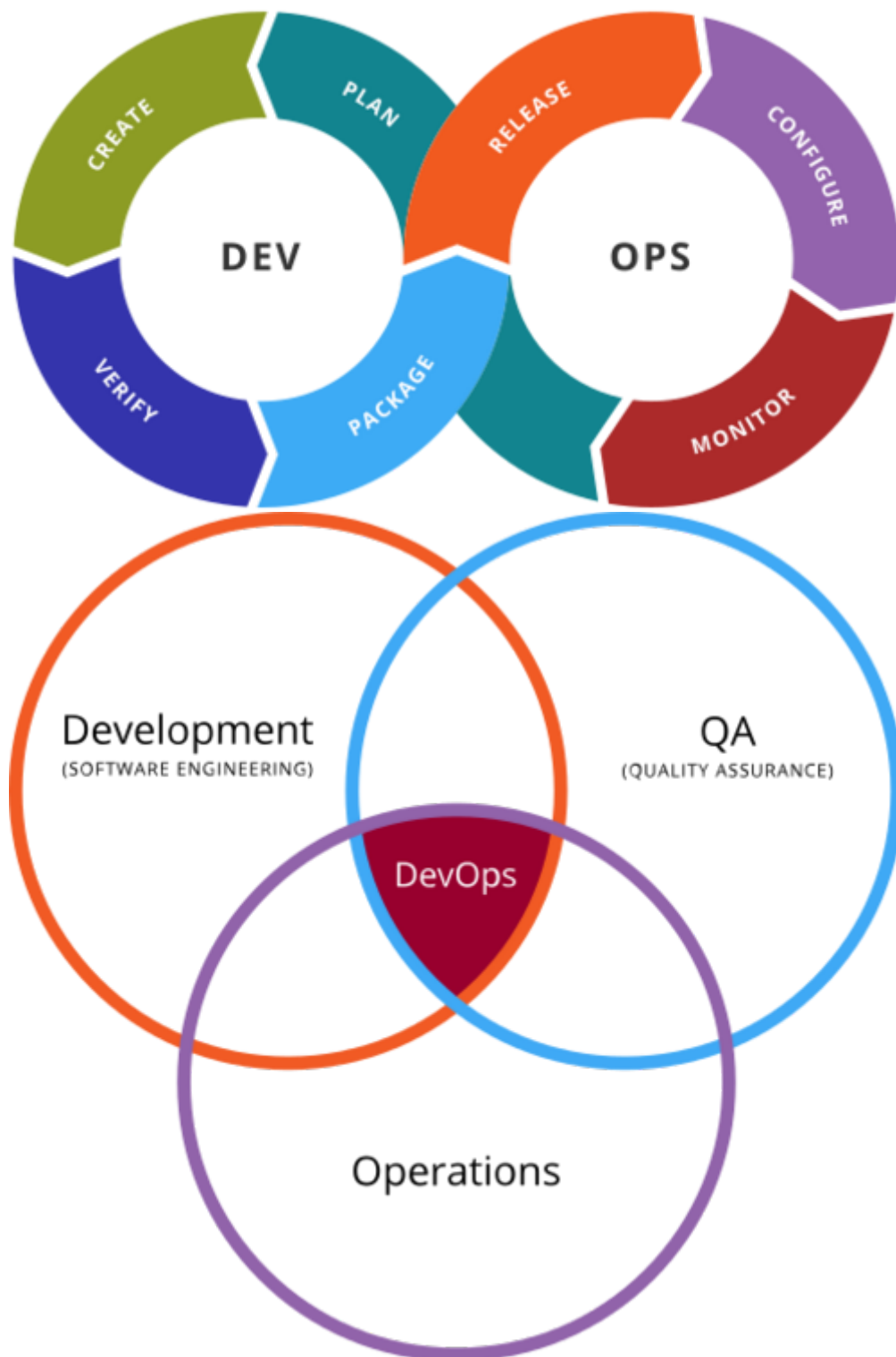
- práctica habitual en desarrollo ágil
- se aplican cambios de forma continua
- se automatiza integración: compilación y pruebas entre componentes
 - no solo en el componente, si no en la interacción entre ellos
- objetivo: detectar pronto los errores
- aplicaciones utilizadas:
 - control de versiones: GIT, SVN, ...
 - Software CI: jenkins, bamboo, travis, ...

Objetivos y responsabilidades

- desarrolladores (devs) que quieren pasar a producción todas las modificaciones
- administradores (ops) que quieren mantener el sistema funcionando sin errores

DevOps

- Desarrolladores
- Testadores (Q&A)
- Operadores



Entrega y despliegue continuo

- Más allá de CI
 - entrega continua : Integración continua, despliegue manual
 - entrega continua + despliegue continuo = todo automatizado
- Se automatiza la generación de una versión publicable (entrega continua)

- se automatiza su paso a producción (despliegue continuo)
- incluye en el ciclo a Q&A y Ops

Aplicaciones utilizadas

- Control de versiones
- herramientas de empaquetado
- orquestación de escenarios
 - los escenarios de desarrollo y producción han de ser iguales (para la correcta comprobación de la aplicación)
- automatización de la configuración
 - herramientas de configuración de la aplicación: crea/replica el escenario necesario para la app

¿devops sin devs?

- ¿podemos tratar la configuración de sistemas como código?
 - si, ya no hacemos el proceso en la máquina, desarrollamos el código que generará esa máquina con sus requisitos
 - desarrollo el escenario que despliega las configuraciones de nuestro sistema
- ¿un administrador que sólo programa y monitoriza?
- Administramos sin tocar en producción
 - administrador de sistemas: ejemplo: aplicación de parche de seguridad → aplico los cambios en mi entorno de desarrollo, veo que eso no rompe la aplicación → se aplica en producción
- ¿desarrolladores de sistemas?
 - metodologías ágiles
 - trabajo en equipo
 - desarrollo
 - cambio de paradigma

Site Reliability Engineer (SRE)

- no necesariamente integrado con devs
- centrado en el entorno de producción
- gestión automática de infraestructura
- actualizaciones
- respuesta a eventos
- <https://devops.com/sre-vs-devops-false-distinction/>

Lenguaje de marcas

ocultar/mostrar

¿no es mejor usar word?

- los formatos binarios con «informática de los 90s», su objetivo no declarado en la incompatibilidad
- formatos en texto plano
- legibles tanto por aplicaciones como humanos
- fácilmente modificables
- su éxito radica en la adaptabilidad de aplicaciones y son esenciales hoy en día

XML

- eXtended Markup Language
- estandar de la W3C
- muchos lenguajes que cumplen las reglas

- formato estricto
- documento bien formado
- validación
- esquemas XML (DTD, XML Schema)
 - XML Schema: definición de como ha de ser el XML y puedo validar que un XML cumple con las reglas establecidas
- XML ha quedado restringido a documentos complejos
 - usado en el principio del API RESTFUL

JSON

- JavaScript Object Notation
- Más sencillo que XML
- Sin esquema y no validable
- Muy extendido en APIs REST

YAML

- YAML Ain't Markup Language
- Más sencillo aún que JSON
- Relacionado con definición de aplicaciones sencillas
- Actualmente es el formato «de moda»

HSL

Ejemplos

```
<?xml version="1.0" encoding="ID0-8859-1"?>
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada de Laurentis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
</bookstore>
```

```
{ books: [
  {
    "title": "Everyday Italian",
    "author": "Giada de Laurentis",
    "year": "2005",
    "price": "30.00"
  }
]
```

```
books:
-
  title: "Everyday Italian"
```

author: "Giada de Laurentis"
year: 2005
price: "30.00"

Lenguaje de programación más usados

ocultar/mostrar

Compilados VS interpretados

- en sistemas, típicamente interpretados
- más sencillos
- fácilmente modificables
- pero menos eficientes
- sobre gustos... colores

Algunos lenguajes de programación

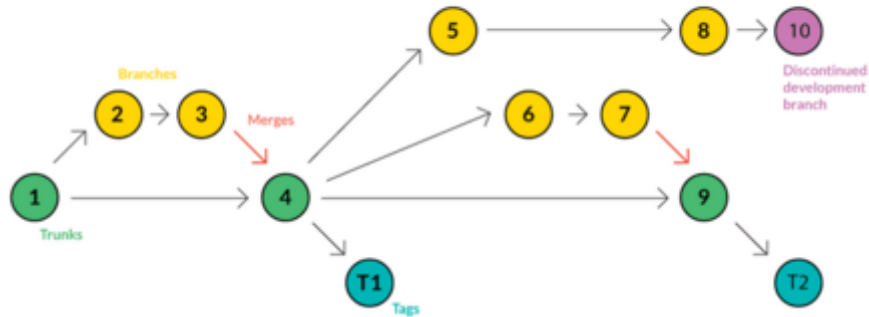
- ¿bash?
 - pobre en estructuras
- Perl
 - cayendo en desuso
 - sintaxis compleja, difícil captar usuarios
- Python
 - simpleza en sintaxis
 - powerful
 - fácil reutilización de código (o encontrar biblioteca)
- Ruby
 - estilo python
- Nodejs
 - Google
- Go
 - compilado
 - en discusión si es el más adecuado para sistemas
 - la tendencia en las últimas herramientas en este sentido es con este lenguaje
- Rust
 - en desarrollo

Sistema de Control de Versiones

ocultar/mostrar

Adoptar las buenas prácticas de los desarrolladores para sistemas

What is "version control"?



Sistemas de control de versiones

- controla cada versión de los ficheros
- muy extendido en desarrollo de software
- esencial en el trabajo colaborativo
- gestiona código fuente, no binarios
- se utiliza también en documentación
- ¿en sistemas? ¡por supuesto!

Implementaciones

- CVS
- SVN
- Bazaar
- Mercurial
- Git

Git

- distribuido
- creado en 2005 por Linus Torvalds
 - para el kernel Linux
- GPL2
- terminología
 - repo
 - commit
 - guardado de versión en local
 - clone
 - push
 - envío a repositorio «principal»
 - branch
 - checkout
 - tag
 - merge

Github

- forja para alojar proyectos
- sin límite en repositorios públicos
- promueve la promiscuidad
 - no necesariamente al estar en GitHub implica soft libre → licencias!
- interfaz web sencilla
- muy importante en software libre
- alojamiento de proyectos personales
- ¡Hazme un fork cariño!

- nuevo repositorio a partir de otro en un determinado momento
- Pull-requests
 - desde el fork, se añade una funcionalidad que se pasa al repositorio original (y este puede o no aceptar)
 - ansible (RedHat)

GitLab

- tuvieron un crash y mostraron en directo la recuperación

Ejemplo de Git

ocultar/mostrar

```
ssh-add ~/.ssh/clave.privada
# añadimos nuestra clave privada para autenticarnos contra github
git clone git@github.com:albertomolina/openstack-ubuntu-ansible.git
# descargamos el repositorio vía SSH
git status
git diff <archivo>
```

Sistemas de aprovisionamiento

ocultar/mostrar

Aprovisionamiento de recursos

- forma parte de un concepto más amplio: orquestación
- esencial en cloud computing

Aprovisionamiento fácil

- creación de escenarios sencillos para demos: Vagrant
- Vagrant utiliza por defecto Virtualbox
 - puede usar varios, incluso cloud
- plantillas sencillas

Aprovisionamiento "para mayores"

- AWS Cloudformation
 - específica y exclusiva para AWS
 - plantillas en formato JSON
- OpenStack Heat
 - desplegar plantillas
 - reutiliza plantillas de Cloudformation
 - formato .hot (YAML)
- Hashicorp Terraform
 - 2 años
 - hermano mayor de Vagrant
 - plantilla independiente del proveedor: AWS, OpenStack, Azure (en general)
 - formato HSL (formato propio)
- OASIS TOSCA

- desarrollo formato de documentos
- no ha conseguido extenderse

Ejemplo de Vagrant

ocultar/mostrar

- fichero de plantilla en formato Ruby
 - `vagrant init` → crea fichero «base»
- `vagrant up` → crear escenario
- `vagrant box list`
 - descargables desde Hashicorp (Atlas imagenes de Vagrant)
- `vagrant ssh`
 - las versiones más viejas de imágenes de vagrant llevan incorporada una par de claves para acceder por SSH sin contraseñas
 - las nuevas tienen una para cada máquina

Ejemplo de Terraform

ocultar/mostrar

- terraform no está aún en distribución de paquetes (descarga directa)
- `terraform -version`
- ficheros `.tf` en formato HLS
- `terraform plan`
- EC2 Dashboard > instances
- `terraform apply`
- `terraform show`
- herramienta de gestión de la configuración del despliegue:
 - `ansible`
 - `puppet`
 - `chef`

Sistema de la gestión de la configuración

ocultar/mostrar

Aprovisionamiento de recursos

- Configuration Management Software
- Base de la Infraestructura como código
- puede realizar aprovisionamiento, pero se especializa en la configuración

Aplicaciones más usadas

- CFEngine
- Puppet (manifiestos - Manifests)
 - agentes (aunque es posible trabajar de otra manera)
 - gestión desde un nodo central. Comunicación bidireccional
- Chef (recetas - Recipes)

- agentes (aunque es posible trabajar de otra manera)
- gestión desde un nodo central. Comunicación bidireccional
- Ansible (Libros de jugadas - Playbooks)
 - RedHat
 - sin agentes, conexiones SSH
 - sintaxis plantillas más sencilla
- Salt (estados - States)
 - sin agentes, sin SSH... cola de mensajes
 - sintaxis plantillas más sencilla
- Juju (Encantamientos - Charms)
 - Canonical
 - utilizado campo específico (Ubuntu), no de uso general

Ejemplo Ansible

ocultar/mostrar

```
ssh-add private.key
# añade la clave privada para no tener que mencionarla explícitamente
```

¿concepto entorno virtual?

ansible.cfg:

- indicamos el fichero con las direcciones de los servidores que queremos gestionar con Ansible (ansible_hosts)
- clave privada (si corresponde)

ansible_hosts:

- secciones con datos de servidor:

```
[back-end-servers]
database ansible_host=10.0.0.1 ansible_port=22 ansible_user=postgres
webserver ansible_host=10.0.0.2 ansible_port=22 ansible_user=root

[infrastructure]
ldap ansible_host=10.0.0.100 ansible_port=22 ansible_user=root
```

Modo de funcionamiento de Ansible:

- línea de comando (simple)
 - `ansible all -m ping` : usa el módulo ping para ver si todos los servidores están ON
 - `ansible all -m apt -a update_cache=yes --sudo (o -s)` : ejecuta los procesos solicitados
 - respuestas
 - verdes : OK
 - amarillas: cambios en la máquina (no necesariamente malo)
 - rojas: errores
- Playbooks (recetas)
 - idempotencia: propiedad matemática que al aplicar una función sobre un objeto para que siempre le de lo mismo

- le digo a que estado quiero llegar (quiero tener instalado apache) y Ansible realiza los pasos necesarios (si está instalado, responderá en verde que ya está, en amarillo si lo ha instalado, en rojo si no ha podido)
- formato YUML
- `ansible-playbook <fichero_playbook.yml> -s`
`# -s de sudo`

From:

<https://matebcn.eu/> - **miguel angel torres egea**

Permanent link:

<https://matebcn.eu/info:cursos:openwebinars:intro-cloud-computing:iaac>

Last update: **14/09/2018 11:41**

