

INFRAESTRUCTURA COMO CÓDIGO

JOSÉ DOMINGO MUÑOZ

IES GONZALO NAZARENO

MAYO 2022



INFRAESTRUCTURA COMO CÓDIGO. ANSIBLE Y VAGRANT



EVOLUCIÓN DE LA GESTIÓN DE LA INFRAESTRUCTURA



- Aprovisionamiento del servidor:
 - ▶ Comprar el servidor o crear la máquina virtual
 - ▶ Instalar y configurar el SO
 - ▶ Instalar y configurar los servicios
 - ▶ Configurar la seguridad
- Despliegue de aplicaciones
- Documentar todo es la clave. ¿Se documenta?
- Misma configuración utilizada por años
- Escalado vertical, que implica paradas del servidor



DESPLIEGUE “MODERNO” DE UN SERVIDOR

- Aparición del Cloud Computing (IaaS): AWS, Azure, GEC, OpenStack, ...
- Aprovisionamiento de una máquina virtual o contenedor desde una imagen o plantilla
- Uso de herramientas de gestión de la configuración:
 - ▶ Configuración del SO
 - ▶ Instalación y configuración de servicios
 - ▶ Configuración de la seguridad
 - ▶ Actualizaciones
- Despliegue de aplicaciones desde un entorno de pruebas idéntico al de producción
- Idealmente se utiliza escalado horizontal
- Los servidores no tienen por qué mantener la misma configuración mucho tiempo



INFRAESTRUCTURA COMO CÓDIGO



¿PODEMOS PROGRAMAR LA INFRAESTRUCTURA?

- Podemos programar la creación y mantenimiento de escenarios virtualizados y en el cloud.
 - ▶ Los servicios de virtualización y de Cloud Computing (IaaS) son software, por lo tanto los podemos programar. **¿Cómo? APIs!!!**
 - ▶ Los programas que nos permite programar la creación de escenarios lo llamamos: **Software de Orquestación.**
- Podemos programar la configuración de los servicios y aplicaciones que corren en las máquinas (de los escenarios creados).
 - ▶ Los programas que nos ayudan a configurar el software de la máquinas lo llamamos: **Software de gestión de la configuración (CMS)**



- Utilizado para crear escenarios completos con múltiples servidores o contenedores (aprovisionamiento de recursos).
- Muy útil en demanda variable de recursos
- Muy útil en entornos en los que se cambia continuamente la configuración
- Puede incluir funcionalidad de autoescalado
- Pueden incluir repuestas a eventos

Ejemplos: **Vagrant, Terraform, Cloudformation (AWS), Heat(OpenStack), Juju, ...**



Nos proporciona la gestión e instalación (de forma **declarativa**) del software de la infraestructura que hemos creado:

- Nos permite instalar software de forma automática,
- Se utilizan ficheros de texto donde **declaramos** la configuración que deseamos obtener.
- Se pueden usar para gestionar las actualizaciones de los sistemas.
- Idempotencia: Podemos ejecutar una declaración múltiples veces y el resultado tiene que ser siempre el mismo.

Ejemplos: **Puppet, Chef, Ansible, Salt, ...**



¿HAY DIFERENCIAS ENTRE ORQUESTACIÓN Y GESTIÓN DE LA CONFIGURACIÓN?

- Normalmente, el software de Orquestación también nos permite realizar configuración automática.
- Normalmente, el software de Gestión de la Configuración también nos permite realizar Orquestación.

¿No hablamos de lo mismo?

¿Hay diferencias entre crear un escenario y configurar el software de la infraestructura creada?



Usa tu infraestructura como el software que es:

- Utiliza software de control de versiones
- Utiliza un buen editor de textos
- Todo legible y con comentarios
- Utiliza software de orquestación
- Utiliza software de gestión de la configuración
- Devops



- **Conflicto:** Tradicionalmente **Equipo de Desarrollo (DEV)** y **Equipo de Sistemas (OPS)** han tenido objetivos y responsabilidades diferentes.
- El objetivo debería ser común.
- ¿Cómo solucionarlo?
 - ▶ Mismas herramientas.
 - ▶ Extender buenas prácticas de desarrollo a sistemas: De integración continua a entrega continua o a despliegue continuo.
 - ▶ Infraestructura como código
 - ▶ **Escenarios replicables, automatización de la configuración.**



ANSIBLE



- Es un **Software de gestión de la configuración (CMS)**.
- Nos permite “declarar” la configuración de máquinas, para llevarla a cabo de forma **automática**.
- Desarrollado principalmente por Red Hat: www.ansible.com
- Escrito en Python
- Primera versión: 2012
- Arquitectura push. Se indica la configuración desde la máquina donde está instalado ansible.
- No utiliza ningún agente: ssh
- Jugadas (plays) y libros de jugadas (playbooks) en YAML



¿POR QUÉ ANSIBLE?

- Cualquier CMS es una buena opción
- Ansible es sencillo de aprender y la sintaxis es conocida (YAML)
- Ansible no utiliza agentes, sólo ssh (!)
- Fácil de instalar (disponible en pypi)
- Comunidad muy activa
- Repositorio de playbooks realizados por la comunidad: [Ansible Galaxy](#)
- Más cercano a la forma de trabajar de administradores de sistemas



CONCEPTOS SOBRE ANSIBLE

- **Módulos y plugins:** Distintas funciones predefinidas que nos permiten realizar una acción: copiar un fichero, instalar paquetes, ...
 - ▶ Cada módulo puede recibir parámetros (obligatorios u opcionales).
 - ▶ [Indexes of all modules and plugins](#)
- **Jugada (play):** Declaración en yaml de una acción (se utiliza un módulo) que quiero conseguir.
 - ▶ Ejemplo: Quiero que en la máquina este instalado un paquete de una versión determinada.
- **Libro de jugadas (playbooks):** Conjuntos de jugadas (plays), para conseguir una configuración compleja de la máquina.
- **Roles:** Normalmente dividimos los playbooks por cada servicio que quiero configurar.
 - ▶ Ejemplo: Rol “apache2”: Permite la instalación y configuración de apache2,...
 - ▶ Nos permite la reutilización de código.
- **Variables:** Los plays, normalmente, están parametrizados. Se utilizan variables para concretar la configuración en cada caso.



VAGRANT



- Vagrant
- Aplicación libre desarrollada en ruby que nos permite crear y personalizar entornos de desarrollo livianos, reproducibles y portables.
- Nos permite automatizar la creación y gestión de máquinas virtuales.
- Podemos usar distintos hipervisores: **VirtualBox, VMWare, Hyper-V, KVM, ...**
- **Objetivo principal:** aproximar los entornos de desarrollo y producción, de esta manera el desarrollador tiene a su disposición una manera muy sencilla de desplegar una infraestructura similar a la que se va a tener en entornos de producción.
- A los administradores de sistemas les facilita la creación de infraestructuras de prueba y desarrollo.



- **Boxes:** Un **box** es una imagen de disco que utilizará vagrant para crear las máquinas virtuales.
 - ▶ Se gestionan con el comando vagrant box.
 - ▶ Se trabaja con un **usuario sin privilegio**: los boxes se guardarán en su home.
 - ▶ Repositorio oficial de boxes: [Vagrant Cloud](#)
- **Vagrantfile:** Es el fichero de texto donde declaramos el escenario que queremos construir.
 - ▶ Se declaran: máquinas virtuales, redes, almacenamiento,...
 - ▶ Todas los comandos vagrant se ejecutan en el directorio donde está este fichero.
 - ▶ vagrant up, vagrant halt, vagrant destroy, ...

