ansible

Alberto Molina Coballes

IES Gonzalo Nazareno
CONSEJERÍA DE EDUCACIÓN

31 de enero de 2018

Introducción

Despliegue tradicional en un servidor

- 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
- Misma configuración utilizada por años
- Escalado vertical, que implica paradas del servidor

Despliegue “moderno” de un servidor

- 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

Cambio de paradigma: Infraestructura como código

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 y de gestión de la configuración
- Devops

Software de orquestación

- 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

Software de gestión de la configuración (CMS)

Proporciona la gestión y configuración de:

- Aprovisionamiento
- Instalación
- Configuración
- Actualizaciones

CMS de software libre

Idempotencia

Propiedad de ciertas operaciones matemáticas que pueden aplicarse múltiples veces, cambiando el resultado sólo la primera vez

- Se utiliza en este ámbito para indicar la diferencia entre los CMS y otras opciones de configuración automátia
- Una receta de un CMS se puede ejecutar múltiples veces y el resultado tiene que ser el siempre el mismo

Puppet

- Desarrollado por Puppet Labs: puppet.com
- Escrito en C++ y Clojure
- Primera versión: 2005
- Arquitectura cliente-servidor: Agente en los clientes y servidor con las configuraciones
- Los clientes comprueban cada cierto tiempo si es necesario aplicar cambios y en caso necesario los aplican
- Configuración mediante manifiestos (manifests) con sintaxis propia
- Muchos manifiestos disponibles (Forja de puppet): `https://forge.puppet.com/`

Chef

- Desarrollado por Chef (antes OpsCode): chef.io
- Escrito en ruby
- Primera versión: 2009
- Arquitectura pull: Chef server, chef client y workstation
- Los clientes comprueban cada cierto tiempo si es necesario aplicar cambios y en caso necesario los aplican
- Configuración mediante recetas (recipes) y libros de recetas (cookbooks) en ruby
- Muchos libros de recetas disponibles (chef supermarket): `https://supermarket.chef.io/`

SaltStack (Salt)

- Desarrollado por SaltStack: saltstack.com
- Escrito en Python
- Primera versión: 2011
- Master y minions
- Descripción de recursos en YAML
- Incluye monitorización para respuesta a eventos

Ansible

- Desarrollado principalmente por Red Hat (antes ansible): www.ansible.com
- Escrito en Python
- Primera versión: 2012
- Arquitectura push
- No utiliza ningún agente: ssh
- Jugadas (plays) y libros de jugadas (playbooks) en YAML

¿Por qué ansible?

- Cualquier CMS es una buena opción
- Salt y Ansible son más sencillos de aprender y la sintaxis es conocida (YAML)
- Ansible no utiliza agentes, sólo ssh (!)
- Fácil de instalar (disponible en pypi)
- Comunidad muy activa
- Más cercano a la forma de trabajar de administradores de sistemas

Ansible

Instalación

- `http://docs.ansible.com/ansible/latest/intro_installation.html`
- Está empaquetado para varias distros y hay una PPA para Ubuntu/Debian
- Es muy sencillo hacerlo en un entorno virtual de python con pip

Configuración

- Puede definirse en diferentes ubicaciones, con la siguiente orden de prioridades:
  - `ANSIBLE_CONFIG` (variable de entorno)
  - `ansible.cfg` (directorio actual)
  - `.ansible.cfg` (en el directorio home)
  - `/etc/ansible/ansible.cfg`
- Recomendación: Un repositorio git por configuración con su `ansible.cfg`
- ¿Qué podemos definir aquí? `http://docs.ansible.com/ansible/latest/intro_configuration.html`

Configuración

Ejemplo de configuración:

```
[defaults]
inventory = ansible_inventory.ini
remote_user = debian
private_key_file = /home/user/.ssh/private_key
host_key_checking = False
```

Módulos

- Se pueden ejecutar directamente o configurándolos en libros de jugadas (*playbooks*)
- Cada módulo puede recibir parámetros (obligatorios u opcionales)
- Este enlace: `http://docs.ansible.com/ansible/latest/modules_by_category.html`
- Un ejemplo:

```
ansible controller -m user -a "name=alberto group=adm"
```

Libros de jugadas (*playbooks*)

- Los libros de jugadas agrupan jugadas (*plays*)
- Cada jugada contiene una o varias tareas (y otros elementos)
- Las tareas utilizan módulos
- Se ejecutan secuencialmente
- Están escritos en YAML

- Ansible es fácil de aprender: aprende practicando
- Instálalo, configura un fichero de inventario y practica
- Empieza con tareas sencillas y módulos
- Cuando te familiarices con esos elementos, continúa:
  - Manejadores (*handlers*)
  - Variables: Host, Group, facts, ...
  - Ejecución condicionada
  - Bucles
  - Roles
  - Buenas prácticas y reutilización