

ANSIBLE PARA DEV+OPS

DÍA 1

INTRODUCCIÓN

Sobre versiones de Ansible

¿Para qué es bueno Ansible?

¿Cómo funciona Ansible?

Lo grande de Ansible

Lo no tan grande de Ansible

¿Es ansible demasiado sencillo?

SOBRE VERSIONES DE ANSIBLE

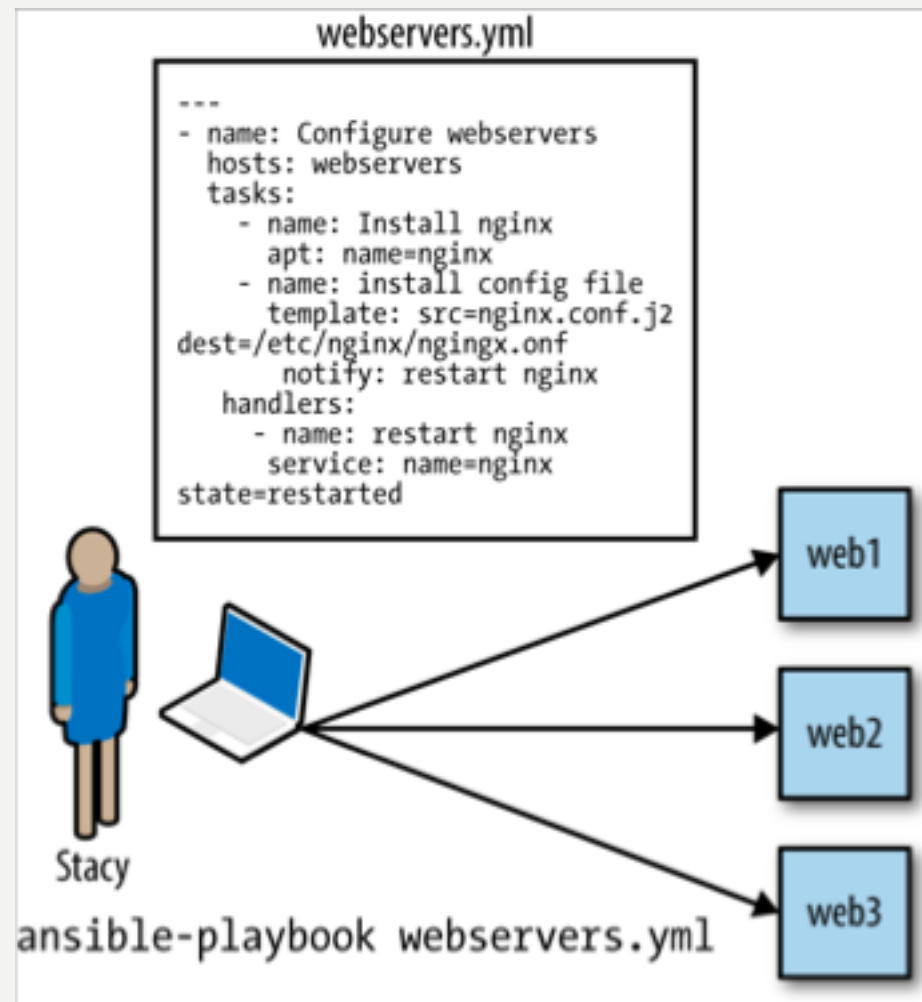
- Este curso está pensado para la versión 2.3.x de ansible
- Los ejemplos funcionarán en versiones anteriores (mayormente) y en posteriores
- Origen del nombre: se tomó del libro El juego de Ender, donde un Ansible era un dispositivo que permitía controlar varias naves remotas a la vez, a distancia

PARA QUE ES BUENO ANSIBLE

- Es una herramienta de gestión de configuración
 - Define estados finales en los que se quiere dejar algún elemento de un servidor
 - Usa un DSL (Lenguaje Específico de Dominio) para describir estos estados
- Es una herramienta de despliegue
 - Permite enviar software generado “in house” a servidores remotos para ponerlo en funcionamiento
- Es una herramienta de orquestación
 - Permite seleccionar en que orden se realizan los pasos
- Es una herramienta de provisioning
 - Permite generar infraestructura en diferentes proveedores de cloud (AWS, Azure, Digital Ocean, GCE, Linode, Rackspace, OpenStack, vSphere...)

COMO FUNCIONA ANSIBLE

- Genera los scripts en python temporales que ejecutan una serie de tareas
- Se conecta por ssh a los nodos donde hay que aplicar estas tareas
- Ejecuta el script en todos los nodos
 - Espera a que cada tarea se ejecute en cada nodo (a no ser que se le diga lo contrario)
 - Ejecuta cada tarea en paralelo en cada nodo
- Espera a que el script se acabe de ejecutar



LO GRANDE DE ANSIBLE

- Sintaxis fácil: los scripts de Ansible (llamados playbooks) se escriben en YAML
- No necesita instalar nada en los hosts remotos: solo Python y la librería simplejson
- Push based: Las nuevas configuraciones se envían del nodo de configuración (la máquina donde se ejecuta Ansible) a los hosts remotos. Esto permite controlar cuando se aplican los cambios de configuración
- Permite adaptar su complejidad al tamaño del despliegue: es igual de potente administrando 1 servidor como administrando 1000.
- Módulos integrados: tiene módulos en su core para **casi** todo (ver http://docs.ansible.com/ansible/latest/modules_by_category.html)

LO NO TAN GRANDE DE ANSIBLE

- Push based: Las nuevas configuraciones se envían del nodo de configuración (la máquina donde se ejecuta ansible) a los hosts remotos. Esto provoca que se tenga que controlar manualmente que versión de cada Playbook se ha aplicado y cuando se ha aplicado.
- Capa muy fina de abstracción: otros sistemas de configuración permiten abstraer conceptos como “package” que engloba los paquetes sea cual sea el origen, Ansible está más orientado a proveer una DSL fácil de implementar y permitir crear scripts (Playbooks) más personalizados, que no sean reutilizables en todos los entornos.
 - Es mejor escribir Playbooks para tu organización que tratar de reaprovechar Playbooks de otros sitios
 - Los Playbooks que encuentres en Github o internet pueden servir de referencia/ejemplo

¿ES ANSIBLE DEMASIADO SIMPLE?

- Depende de SSH para conectar con las máquinas remotas
 - Se puede configurar multiplexing para mejorar el rendimiento (lo veremos más adelante)
- Compatibilidad con Windows limitada
 - Un host Windows no puede ser el control host
- Necesidad de implementar mecanismos de control para saber que versión de que Playbook se ha implementado en un host (o usar Ansible Tower)

CONFIGURACIÓN DE LABORATORIOS

- El laboratorio consta de dos partes:
 - Nodo de control (montado en una Centos 7) a la que accederéis mediante teamviewer
 - Máquinas de prácticas: desplegadas en cloud para cada práctica usando Playbooks de Ansible, se crean y se destruyen bajo demanda en función de cada ejercicio
 - Para ello se os entregará un fichero que os permitirá tener las credenciales necesarias para operar en el cloud

REPOSITORIO DE CÓDIGO

- En el host de control, con el usuario root, ejecutaremos lo siguiente

```
cd && git clone https://gitlab.teradisk.net/trainings/ansible-devops
```

- Copiaremos el fichero `aws_vault.yml` que nos facilitará el formador en `$HOME/ansible-devops/env_vars`

RECORDANDO LO BÁSICO

- Idempotencia
- YAML
- Estructura de un Playbook
 - Plays
 - Tareas
 - Handlers
 - Tags
- Módulos
- Roles
- Inventarios

IDEMPOTENCIA

- Capacidad de realizar una acción determinada varias veces y aún así conseguir el mismo resultado que si se realizara una sola vez
- Crítica para el buen funcionamiento de un sistema de gestión de la configuración
- Scripts “home made” -> difícilmente idempotentes
- Playbooks de Ansible -> idempotentes siempre que se usen módulos estándar, para módulos custom hay que escribirlos garantizando idempotencia

YAML

- Lenguaje de markup basado en espacios y tabulaciones
 - Los ficheros empiezan por “---
 - Los comentarios son de una sola linea y empiezan por #
 - Las cadenas no tienen porque estar entre comillas*
 - Los booleanos son True o False
 - Listas:

```
- My Fair Lady  
- Oklahoma  
- The Pirates of Penzance
```

YAML

- Lenguaje de markup basado en espacios y tabulaciones
 - Diccionarios

```
address: 742 Evergreen Terrace  
city: Springfield  
state: North Takoma
```

- Multilinea

```
address: >  
  Department of Computer Science,  
  A.V. Williams Building,  
  University of Maryland  
city: College Park  
state: Maryland
```

ESTRUCTURA DE UN PLAYBOOK

```
- name: Configure webserver with nginx
  hosts: webservers
  become: True
  tasks:
    - name: install nginx
      apt: name=nginx update_cache=yes

    - name: copy nginx config file
      copy: src=files/nginx.conf dest=/etc/nginx/sites-available/default

    - name: enable configuration
      file: >
        dest=/etc/nginx/sites-enabled/default
        src=/etc/nginx/sites-available/default
        state=link

    - name: copy index.html
      template: src=templates/index.html.j2 dest=/usr/share/nginx/html/index.html
              mode=0644

    - name: restart nginx
      service: name=nginx state=restarted
```

ESTRUCTURA DE UN PLAYBOOK

- Un Playbook es una lista de diccionarios a los que llamamos **Plays**.
- Cada Play debe tener
 - Un conjunto de hosts que configurar
 - Una serie de tareas que ejecutar en esos hosts
- Además puede tener:
 - name
 - become
 - vars

```
- name: Configure webserver with nginx
hosts: webservers
become: True
tasks:
  - name: install nginx
    apt: name=nginx update_cache=yes

  - name: copy nginx config file
    copy: src=files/nginx.conf dest=/etc/nginx/sites-available/default

  - name: enable configuration
    file: >
      dest=/etc/nginx/sites-enabled/default
      src=/etc/nginx/sites-available/default
      state=link

  - name: copy index.html
    template: src=templates/index.html.j2
              dest=/usr/share/nginx/html/index.html mode=0644

  - name: restart nginx
    service: name=nginx state=restarted
```


ESTRUCTURA DE UN PLAYBOOK

- Las tareas (Tasks) indican un módulo y su parametrización
- Recomendamos el uso de nombres en las Tasks para mejorar su identificación

```
- name: Pass options to dpkg on run
  apt:
    upgrade: dist
    update_cache: yes
    dpkg_options: 'force-confold,force-confdef'
```

- Existe un tipo de tarea llamada Handler que solo se ejecuta si otra tarea ha realizado cambios en el host (ej: reiniciar servicio si se modifica la configuración)

MÓDULOS

- Los módulos son scripts que vienen con Ansible y son los que realmente hacen acciones en un host.
- Los más habituales:
 - Apt/yum
 - Copy
 - File
 - Service
 - Template
- Para ver la documentación podemos ir a la web de Ansible (<http://docs.ansible.com/ansible/latest/>) o usar el comando ansible-doc

ROLES

- Permiten agrupar tareas repetitivas para poderlas reutilizar en diferentes Playbooks
- Pueden tener dependencias entre ellos, valores por defecto...
- Al ser unidades de desarrollo más pequeñas permiten tener una mejor mantenibilidad

INVENTARIOS

- Permiten tener identificados grupos de máquinas por categorías, tags
- Permiten asignar variables en función de los tipos de máquinas
- Pueden ser ficheros estáticos o scripts que devuelvan la información en base a una fuente de datos (lo veremos en el día 3 del curso)

LABS

- Inicializar máquinas de labs “dia I -lab I”
- Crear el inventario estático de las 3 máquinas que se creen
- Crear un Playbook que instale Nginx en las 3 máquinas y despliegue una página de bienvenida
- Modificar la página de bienvenida en función del host donde se ejecuta
- Segregar el Playbook mediante roles
 - Rol nginx
- Destruir máquinas de labs “dia I -lab I”

LABS

- Inicializar máquinas de labs “dia1-lab2”
- Crear el inventario estático de las 3 máquinas que se creen de forma que una sea el grupo database y las otras dos el grupo www
- Crear un playbook que instale nginx en las máquinas www y despliegue una página de bienvenida, y que instale mysql en las máquinas bbdd
- Segregar el playbook mediante roles
 - Rol nginx
 - Rol mysql
- Destruir máquinas de labs “dia1-lab2”