ANSIBLE PARA DEV+OPS

DÍA 3

LO QUE VEREMOS HOY

- Inventario dinámico
- Templating avanzado
- Ansible Galaxy
- Escritura de módulos custom
- Labs:
  - Templates con control de flujo e inclusión
  - Módulo custom
  - Inventario dinámico desde una CMDB

INVENTARIO DINÁMICO

- Concepto
- Interfaz mínimo
- Ejemplos de salidas
- Ejemplos de inventarios dinámicos existentes

CONCEPTO

- En grandes entornos seguro que existe una CMDB que almacena todos los datos de los servidores gestionados
- No tiene sentido mantener manualmente los ficheros por clientes si existe una fuente dinámica de datos
- Ansible tiene un interfaz mediante el cual puede usar scripts para crear esos inventarios

INTERFAZ MÍNIMO

- --host=<hostname> debe devolver el detalle de un host

```
{ "ansible_host": "127.0.0.1", "ansible_port": 2200,
  "ansible_user": "vagrant"}
```

- --list debe listar los grupos
- La salida es un json con las variables

INTERFAZ MÍNIMO

- --list debe listar los grupos
- La salida es un json con las variables

```
{"production": ["delaware.example.com", "georgia.example.com",
                "maryland.example.com", "newhampshire.example.com",
                "newjersey.example.com", "newyork.example.com",
                "northcarolina.example.com", "pennsylvania.example.com",
                "rhodeisland.example.com", "virginia.example.com"],
 "staging": ["ontario.example.com", "quebec.example.com"],
 "vagrant": ["vagrant1", "vagrant2", "vagrant3"],
 "lb": ["delaware.example.com"],
 "web": ["georgia.example.com", "newhampshire.example.com",
         "newjersey.example.com", "ontario.example.com", "vagrant1"]
 "task": ["newyork.example.com", "northcarolina.example.com",
          "ontario.example.com", "vagrant2"],
 "rabbitmq": ["pennsylvania.example.com", "quebec.example.com", "vagrant3"],
 "db": ["rhodeisland.example.com", "virginia.example.com", "vagrant3"]
}
```

INTERFAZ MÍNIMO

- La salida es un json con las variables
- Ver ejemplo -> vagrant.py

https://github.com/ansible/ansible/blob/devel/contrib/inventory/vagrant.py

TEMPLATING AVANZADO

- Jinja2
- Funciones de Jinja2
- Control de flujo en Jinja2

JINJA2

- Motor de templating de python
- Inicialmente diseñado para páginas web
- Variables entre {{ }}
- Estructuras de control entre {% %}
- Las variables se procesan mediante filtros {{ variable | filtro }} o por métodos {{ variable.método }}

FUNCIONES DE JINJA2

- http://jinja.pocoo.org/docs/2.9/templates/
- http://docs.ansible.com/ansible/latest/playbooks_filters.html
- Las más usadas
  - upper / lower
  - to_json / to_yaml
  - random
  - regex

ESTRUCTURAS DE CONTROL

- {% if <condition> %} xxx {% elif <condition> %} xxxx {% else %} xxxx {% endif %}
- For

```
{% for user in users %}
    <li>{{ user.username|e }}</li>
{% else %}
    <li><em>no users found</em></li>
{% endfor %}
```

MACROS

- Las macros son funciones, se definen entre {% macro nombre(variables) -%} xxxxx {%- endmacro %}

```
{% macro input(name, value='', type='text', size=20) -%}
    <input type="{{ type }}" name="{{ name }}" value="{{
        value|e }}" size="{{ size }}">
{%- endmacro %}
```

- Y luego se llaman

```
<p>{{ input('username') }}</p>
<p>{{ input('password', type='password') }}</p>
```

INCLUDES

- Permiten incluir ficheros de template dentro de ficheros de template para mejorar la legibilidad del código

```
{% include "files/" + item.documentrootdir + ".conf" ignore missing %}
```

NOTA FINAL

- Toda variable de ansible entre comillas “” o apostrofes ‘’ es tratada como un template de jinja2
- Esto permite usar lógica compleja más allá del when

```
esb_master_node: "{% for host in groups['security_group_' + NODE_ENV + '_esb_master'] %}{{ hostvars[host]['ansible_hostname'] }}{% endfor %}"
```

ANSIBLE GALAXY

- ¿Que es?
- ¿Hasta donde abarca?
- Ejemplos de playbooks útiles

QUE ES ANSIBLE GALAXY

- Repositorio de roles de la comunidad que tienen objetivosque pueden considerarse comunes, pero no lo suficiente como para ser módulos internos de ansible
- Cualquiera puede contribuir
- La comunidad puntúa los roles
- https://galaxy.ansible.com/explore#/

HASTA DONDE ABARCA

- La calidad de los roles es variable
- Algunos de los roles se dejan de mantener
- Es importante entenderlos antes de usarlos

PLAYBOOKS ÚTILES

- Ansistrano
- Elasticsearch
- dev-sec.*

- Todos son útiles para aprender como hacen el trabajo en otros sitios
- MINILAB -> ver los roles https://github.com/geerlingguy/ansible-role-redis y https://github.com/elastic/ansible-elasticsearch

MÓDULOS CUSTOM

- Que son
- Cuando son útiles
- Ejemplo: comprobar que podemos llegar a un servidor remoto

QUE SON LOS MÓDULOS CUSTOM

- Permiten crear tareas que son demasiado complejas para los módulos shell o command
- Ejemplos
  - Interactuar con APIs internas de nuestra organización
  - Conseguir IP pública cuando estamos tras de nat
  - Configurar alguna aplicación interna de nuestra organización

CUANDO SON ÚTILES

- Cuando los módulos command / shell se quedan cortos
- Cuando el módulo script se vuelve demasiado complejo
- Cuando no podemos garantizar la idempotencia usando command / shell / script

EJEMPLO: CAN_REACH

- En script

```
- name: run my custom script
  script: scripts/can_reach.sh www.example.com 80 1
```

```
#!/bin/bash
host=$1
port=$2
timeout=$3

nc -z -w $timeout $host $port
```

EJEMPLO: CAN_REACH

- En módulo

```
- name: check if host can reach the database server
  can_reach: host=db.example.com port=5432 timeout=1
```

SOBRE LOS MODULOS CUSTOM

- Ansible los buscará en el directorio library relativo al Playbook
- Ansible espera la salida en formato json:

```
{'changed': false, 'failed': true, 'msg': 'could not reach the host'}
```

- Ansible prevé la clase ansible.module_utils.basic.AnsibleModule para preparar módulos en python (se verá mas en el lab)

LABS

- Crear el lab “dia3-lab1”
- Estudiar el código del rol “laboratiorio-jinja2” y su template
- Aplicarlo
- Validar resultado
- Crear un filtro custom de Jinja para simplificar el template
- Destruir el lab

LABS

- Crear el lab “dia3-lab1”
- Analizar el código del Playbook “custommodule”
- Crear un módulo que reemplace los chequeos que se hacen mediante “command” en el Playbook
- Aplicar el Playbook
- Destruir el lab

LABS

- Desplegar el Playbook “datasource-inventory” en la máquina de laboratorio
- A partir de ese momento, la llamada a http://ip/inventory muestra una estructura en json con los datos de los servidores
- Generar un script de inventario dinámico que Ansible pueda usar basado en la llamada a esa web