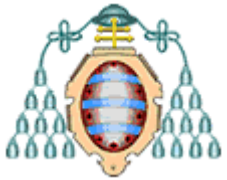
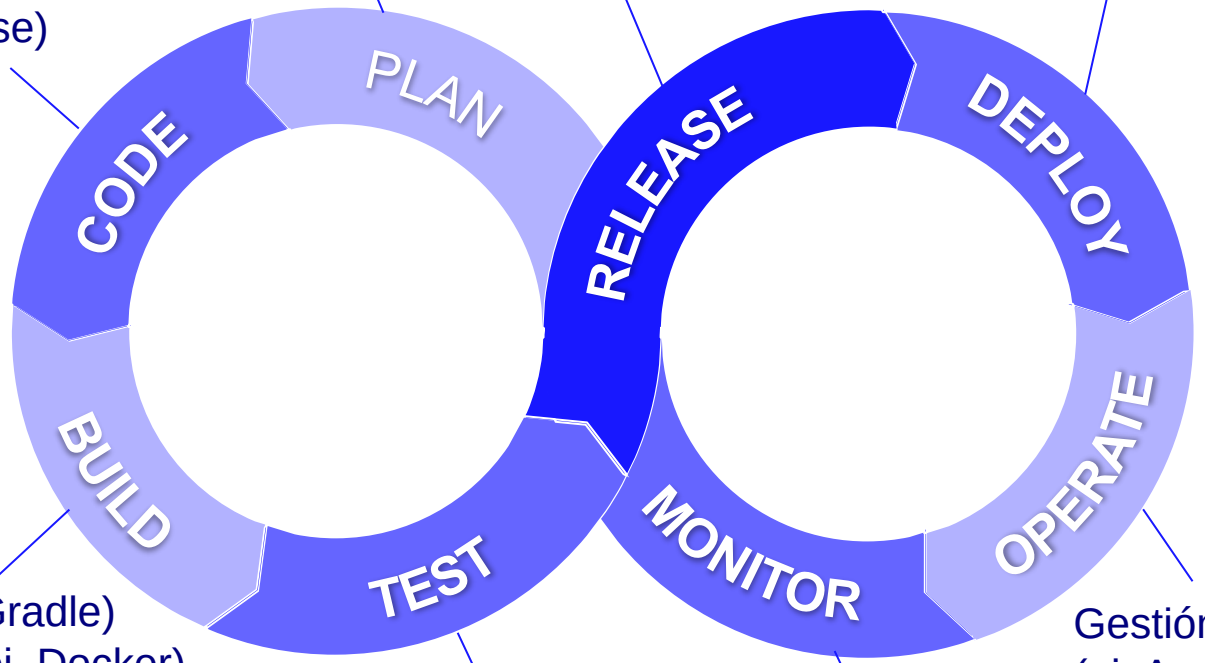
Infraestructura como código

Jesús Morán

Grupo de Investigación en Ingeniería del Software
http://giis.uniovi.es
Universidad de Oviedo



DevOps



Aprovisionamiento (ej. Terraform)
Gestión de configuración (ej. Ansible)
Orquestación (ej. Kubernetes)

Gestión de proyectos (ej. Trello)
Tickets (ej. Jira)

Integración continua (ej. Jenkins)

IDEs (ej. Eclipse)
VCSs (ej. Git)

Build Tools (ej. Gradle)
Contenedores (ej. Docker)

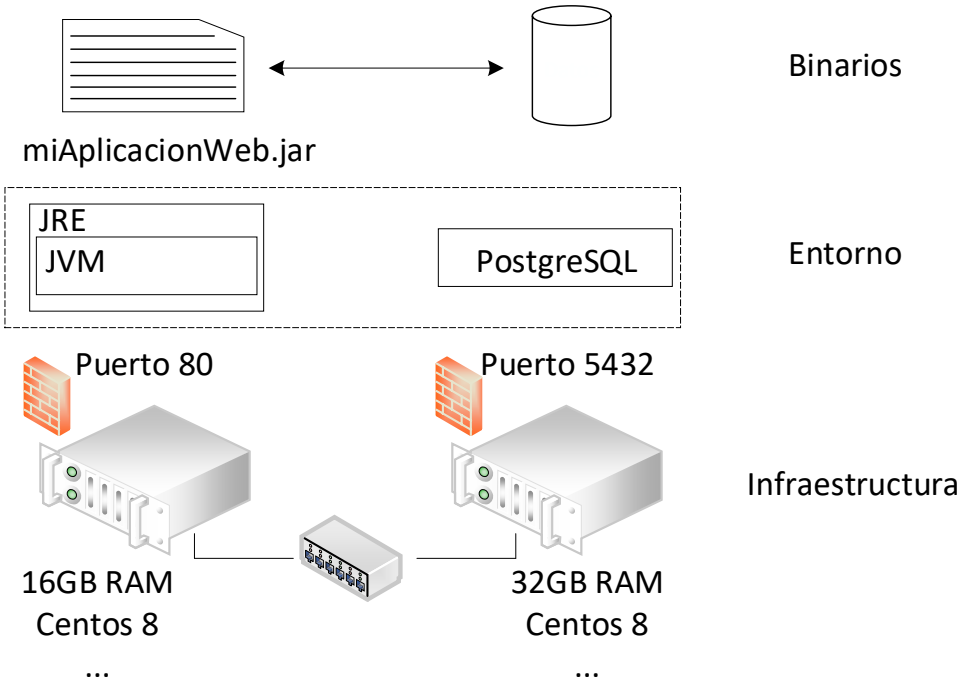
Estáticas (ej. Sonarqube)
Dinámicas (ej. XUnit, Selenium)

Monitorización IT (ej. Nagios)
Monitorización de datos (ej. Splunk)

Gestión de configuración
(ej. Ansible)

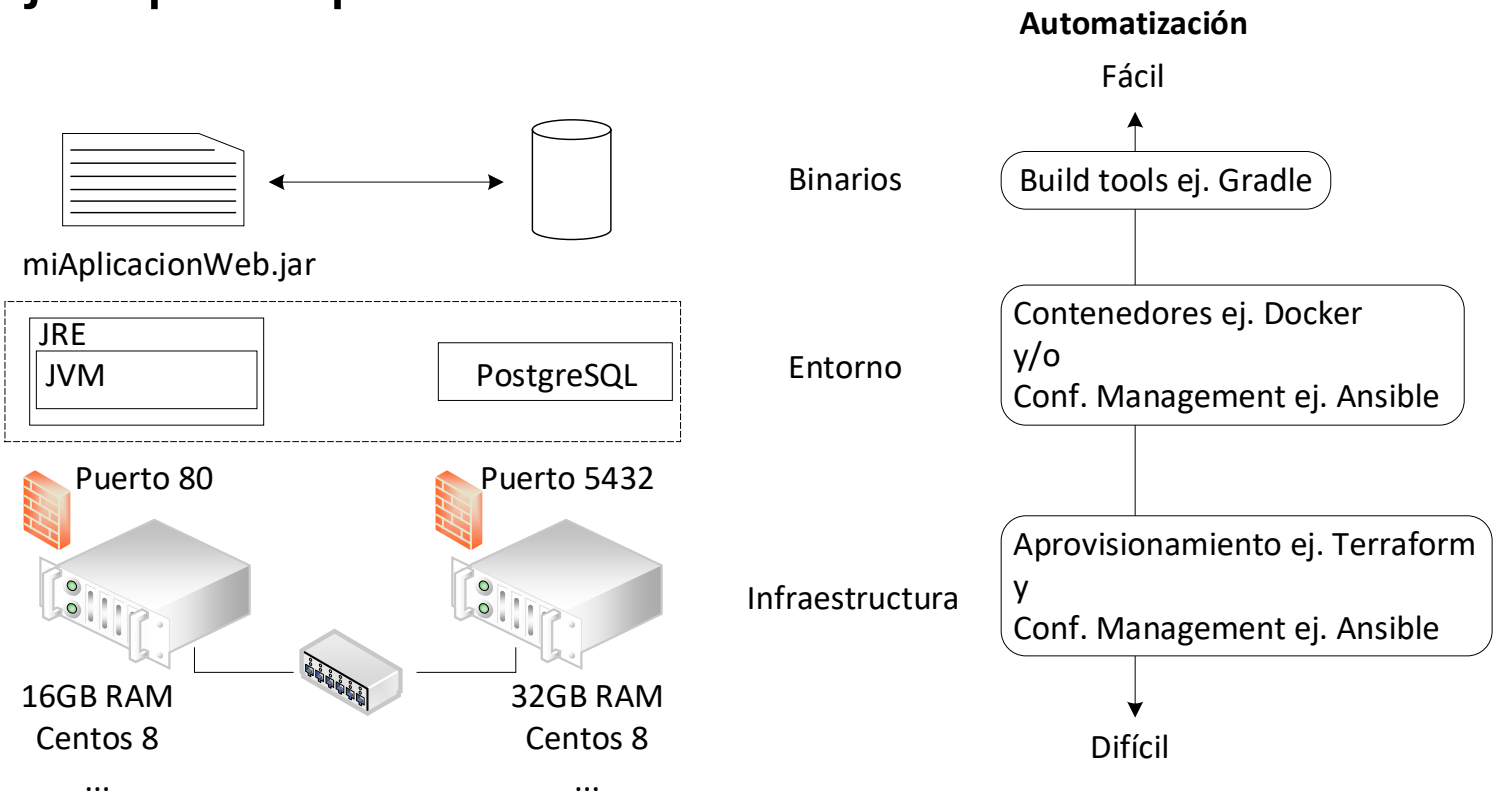
Programa

- Ejemplo: aplicación web



Programa

- Ejemplo: aplicación web



Infraestructura como código

- Código que obtiene la infraestructura:
  - Aprovisionamiento
    - Necesito una máquina de 16GB, otra de 32GB y un router al que están conectadas
    - Herramientas: Terraform (cloud), Vagrant (VM), MaaS (bare-metal)…
  - Gestión de la configuración:
    - Necesito un JRE con la variable de entorno JAVA_HOME, el puerto 80 abierto,…
    - Herramientas: Ansible, Chef, Puppet,…

Infraestructura como código

- Ventajas:
  - Automatizar el despliegue
  - Replicar la infraestructura ej. preproducción, pruebas
  - Reutilizar códigos de infraestructura de terceros
  - Documentar la infraestructura
  - Infraestructura consistente y predecible

- Desventajas:
  - Se tienen que utilizar más tecnologías
  - "*If you automate a mess, you get an automated mess*"
    Rod Michael

Infraestructura como código

- Paradigmas
  - **Mutable**: Adaptar la infraestructura existente
    - Ventajas:
      - Enfoque tradicional
    - Desventajas:
      - Suele mezclarse con cambios manuales (Configuration drifts)
      - Difícil de documentar todos los cambios
      - Infraestructura frágil después de varios cambios
      - Es difícil replicar la infraestructura ej. pruebas

Infraestructura como código

- Paradigmas
  - Mutable: Adaptar la infraestructura existente
  - **Inmutable**: Se crea nueva infraestructura
    - Ventajas:
      - Consistencia y predictibilidad
      - Despliegues atómicos: o se despliega toda la infraestructura o nada
      - Roll backs sencillos a la anterior versión
      - Fácil de replicar infraestructura ej. pruebas
    - Desventajas:
      - Gestionar datos ej. logs, etc (se suele externalizar)
      - Se requiere un stack de tecnologías de automatización

Infraestructura como código

- Lenguajes:
  - Imperativos: indicamos los pasos para conseguir la infraestructura
    - Ej. conéctate a la máquina CentOs, crea un usuario, descarga PostgreSQL, arranca el servicio, etc
    - Ventajas: enfoque tradicional con Shell/bash scripts
    - Inconvenientes: no se adapta cuando cambia el estado inicial
  - Declarativos: indicamos cómo es la infraestructura
    - Ej. quiero que la máquina CentOs tenga un usuario y PostgreSQL
    - Ventajas: es intuitivo para Infraestructura como Código
    - Inconvenientes: incita a utilizar las opciones por defecto

Aprovisionamiento

- Terraform
  - Aprovisiona una infraestructura automáticamente
  - Lenguaje declarativo
  - Multi-Cloud
  - ¿Cloud agnóstico?
  - Idempotente
  - Proceso:
    - Se escribe una plantilla con la infraestructura: HCL
    - Plan: permite ver qué es lo que pasará si se ejecuta
    - Apply: provisiona la infraestructura

Aprovisionamiento

- Terraform

```
provider "aws" {
  acces_key =  "AK…I"
  secret_key = "Y0…0"
  region = "us-east-2"
}

resource "aws_instance" "myTerraformServer" {
  ami = "ami-02bcbb802e03574ba"
  instance_type = "t3.nano"
  tags = {
    Name = "giis3"
  }
}
```

Aprovisionamiento

- Terraform

1

Proveedores: APIs que permitan subir infraestructura ej. AWS, Azure, Kubernetes,…

https://registry.terraform.io/browse/providers

Es recomendable no ponerlos en el archivo. Alternativas: variables de entorno ó ~/.aws/credentials

```
provider "aws" {
  acces_key =  "AK…I"
  secret_key = "Y0…0"
  region = "us-east-2"
}

resource "aws_instance" "myTerraformServer" {
  ami = "ami-02bcbb802e03574ba"
  instance_type = "t3.nano"
  tags = {
    Name = "giis3"
  }
}
```

Aprovisionamiento

- Terraform

```
provider "aws" {
  acces_key =  "AK…I"
  secret_key = "Y0…0"
  region = "us-east-2"
}

resource "aws_instance" "myTerraformServer" {
  ami = "ami-02bcbb802e03574ba"
  instance_type = "t3.nano"
  tags = {
    Name = "giis3"
  }
}
```

2

Recursos: servidores, redes, servicios, registros DNS, etc. (implementados por los proveedores)

https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance

Nombre para referirse al recurso: aws_instance.myTerraformServer

Multitud de opciones:
https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance

Aprovisionamiento

- Terraform
  - Estimar costes de la infraestructura
    - Sólo para los principales proveedores y recursos
    - Obtiene precio por hora y por mes
    - Requiere tener la versión de pago de Terraform
  - Sentinel: policy-as-code
    - Permite establecer políticas a cumplir ej. sólo se pueden crear máquinas EC2 de tipo t2.nano, siempre que se cree una máquina EC2 tiene que tener la propiedad ipv6_addresses, la infraestructura tiene que tener un coste inferior a 100 euros/mes
    - Requiere tener la versión e pago de Terraform

Aprovisionamiento

- Vagrant
  - Crea y configura máquinas virtuales
    - Puede configurarse con comandos Shell, Ansible, Chef, etc.
  - VagrantFile: se programa en Ruby
  - Varios proveedores:
    - Virtualbox
    - Hyper-V
    - Docker
    - AWS
    - …

Aprovisionamiento

- Vagrant

```
VAGRANTFILE_API_VERSION = 2
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
        config.vm.define "myCentosMachine" do |subconfig|
                subconfig.vm.box = "bento/centos-8.3"

                subconfig.vm.provider "virtualbox" do |v|
                        v.memory = 2048
                        v.cpus = 2
                end
        end
end
```

Sistema operativo (imagen)

Características de la máquina

1. Arrancar máquina: `vagrant up [máquina]`
2. Conectarse: `ssh myCentosMachine`
3. Apagar máquina desde el anfitrión: `vagrang halt [máquina]`
4. Eliminar máquina: `vagrant destroy [máquina]`

Gestión de la configuración

- Ansible
  - Automatiza la configuración de servidores, red, routers, etc
  - Configura Windows y Linux
  - Necesita Python en todos los servidores
  - No requiere agentes: ssh y WinRM
  - Control Node: servidor que tendrá instalado Ansible
  - Managed Node: servidor que se quiere configurar
    - No tiene instalado Ansible
    - Tiene instalado ssh/WinRM con clave pública del Control Node

Gestión de la configuración

- Ansible
  - Inventario: Entornos + IPs/nombres de servidores

```
[ansibleuser@myCentosMachine ~]$ cat /etc/ansible/hosts
[myTestEnvironment]
myArchMachine.local

[myProductionEnvironment]
myCentosMachine.local
myUbuntuMachine.local

[myProductionEnvironment:vars]
ansible_python_interpreter=/usr/bin/python3

[all:vars]
ansible_ssh_private_key_file=~/.ssh/id_rsa_ansible
ansible_become=True
ansible_sudo_pass=myP4SSw0rd
[ansibleuser@myCentosMachine ~]$
```

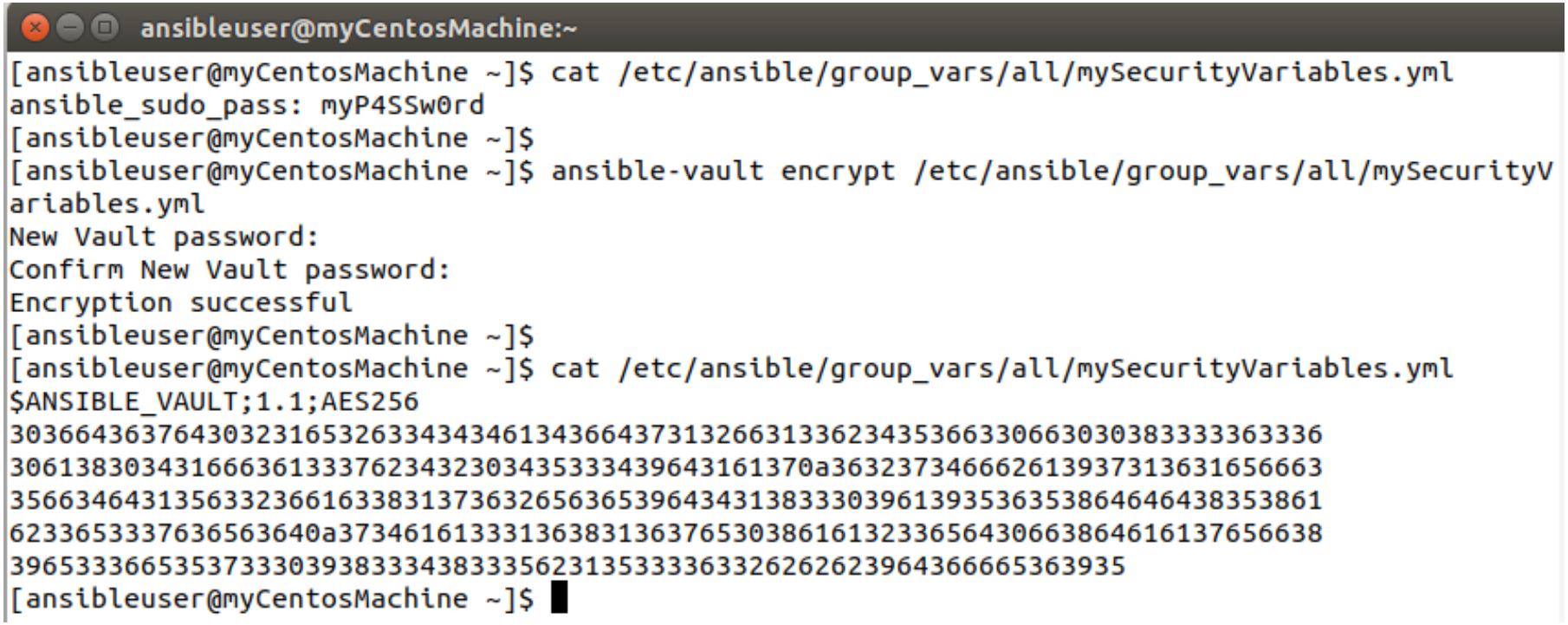
2 entornos y sus servidores

Propiedades para servidores de un entorno

Propiedades para todos los servidores

Gestión de la configuración

- Ansible
  - Inventario: Entornos + IPs/nombres de servidores
    - Vault: encriptación de contraseñas y datos sensibles



Gestión de la configuración

- Ansible
  - Ejecutar un único módulo
    - Ej. `ansible all -m user -a "name=my_user1 create_home=False" --become`

Quiero que todas mis máquinas tengan un usuario “my_user1” sin carpeta home

`all` — Entornos que queremos configurar

`-m user` — Tarea a ejecutar: crear/modificar/eliminar usuarios

`-a "name=my_user1 create_home=False"` — Argumentos:

`--become` — Requiere permisos sudo

name string / required		Name of the user to create, remove or modify. aliases: user
create_home boolean	**Choices:** • no • yes ←	Unless set to `no`, a home directory will be made for the user when the account is created or if the home directory does not exist. Changed from `createhome` to `create_home` in Ansible 2.5. aliases: createhome

El resto toma los de por defecto: ej.

state string	**Choices:** • absent • present ←	Whether the account should exist or not, taking action if the state is different from what is stated.

Gestión de la configuración

- Ansible
  - Ejecutar un único módulo
    - Ej. `ansible all -m user -a "name=my_user1 create_home=False" --become`

Quiero que todas mis máquinas tengan un usuario “my_user1” sin carpeta home

`all` — Entornos que queremos configurar

`-m user` — Tarea a ejecutar: crear/modificar/eliminar usuarios

`-a "name=my_user1 create_home=False"` — Argumentos

```
myArchMachine.local | CHANGED => {
    "changed": true,
    "comment": "",
    "create_home": false,
    "group": 1002,
    "home": "/home/my_user1",
    "name": "my_user1",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 1002
}
```

Crea usuario

```
myCentosMachine.local | CHANGED =>
    "changed": true,
    "comment": "",
    "create_home": false,
    "group": 1002,
    "home": "/home/my_user1",
    "name": "my_user1",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 1002
}
```

```
myUbuntuMachine.local | SUCCESS =>
    "append": false,
    "changed": false,
    "comment": "",
    "group": 1002,
    "home": "/home/my_user1",
    "move_home": false,
    "name": "my_user1",
    "shell": "/bin/sh",
    "state": "present",
    "uid": 1002
}
```

No crea usuario porque ya existe

Gestión de la configuración

- Ansible
  - Ejecutar un único módulo
  - Playbooks: ejecutar varios módulos

Quiero que todas mis máquinas de producción tengan un usuario "myUserPlayBook1" con carpeta home

```
---

- name: This is my first ansible playbook
  hosts: myProductionEnvironment
  become: True
  tasks:
  - name: My first task in the playbook creates an user
    user:
     name: myUserPlayBook1
     create_home: True
     state: present
```

Entorno de producción definido en el inventario

Las máquinas tienen que tener un usuario con carpeta home

Gestión de la configuración

- Ansible
  - Ejecutar un único módulo
  - Playbooks: ejecutar varios módulos

Quiero que todas mis máquinas de producción tengan un usuario “myUserPlayBook1” con carpeta home

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible-playbook myCreateUser.yml

PLAY [This is my first ansible playbook] ***************************************

TASK [Gathering Facts] *********************************************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [My first task in the playbook creates an user] ***************************
changed: [myUbuntuMachine.local]
changed: [myCentosMachine.local]

PLAY RECAP *********************************************************************
myCentosMachine.local      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
myUbuntuMachine.local      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansibleuser@myCentosMachine ~]$
```

Gestión de la configuración

- Ansible
  - Ejecutar un único módulo
  - Playbooks: ejecutar varios r

Quiero que todas mis máquinas de producción tengan un usuario “myUserPlayBook1” con carpeta home y un archivo README.txt que les recuerde que no deben compartir la contraseña

```
---

- name: This is my first ansible playbook
  hosts: myProductionEnvironment
  become: True
  tasks:
  - name: My first task in the playbook creates an user
    user:
     name: myUserPlayBook1
     create_home: True
     state: present

  - name: My second task in the plabook creates a file in the user folder
    copy:
     content: "You must not share your password\n"
     dest: "/home/myUserPlayBook1/README.txt"
```

Copia a la carpeta home un README.txt

Gestión de la configuración

- Ansible
  - Ejecutar un único módulo
  - Playbooks: ejecutar varios r

Quiero que todas mis máquinas de producción tengan un usuario "myUserPlayBook1" con carpeta home y un archivo README.txt que les recuerde que no deben compartir la contraseña

```
---
```

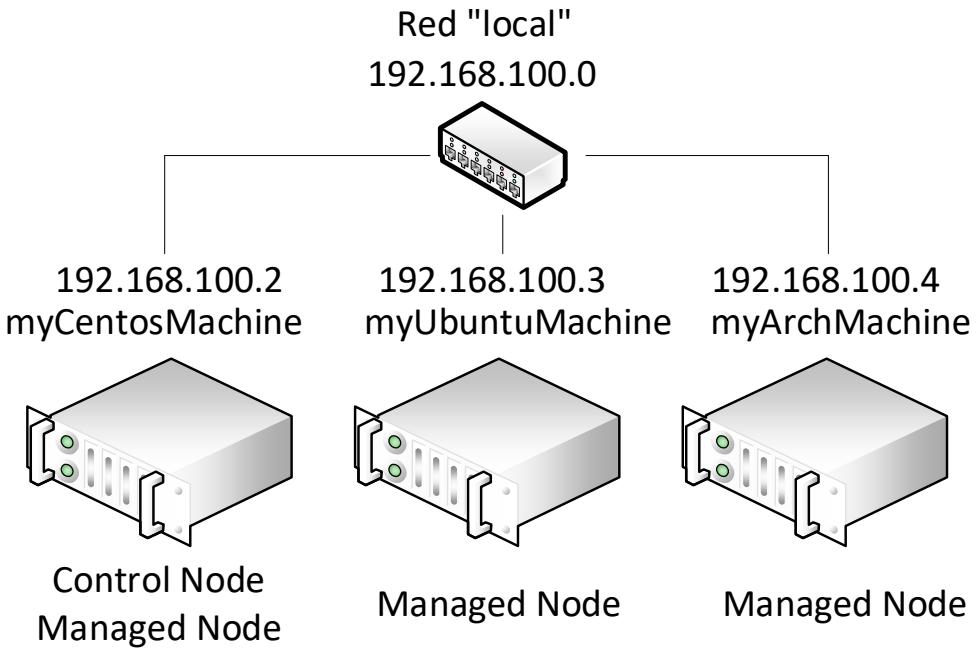
```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible-playbook myCreateUser.yml

PLAY [This is my first ansible playbook] ***************************************

TASK [Gathering Facts] *********************************************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [My first task in the playbook creates an user] ***************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [My second task in the plabook creates a file in the user folder] *********
changed: [myCentosMachine.local]
changed: [myUbuntuMachine.local]

PLAY RECAP *********************************************************************
myCentosMachine.local      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
myUbuntuMachine.local      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansibleuser@myCentosMachine ~]$
```

Gestión de la configuración

- Ansible
  - Ejecutar un único módulo
  - Playbooks: ejecutar varios módulos
    - Roles: agrupaciones de tareas, variables, etc que indican el estado al que tiene que llegar una máquina. Ej. rol de PostgreSQL
    - Galaxy: repositorio de roles

Tutorial Ansible

- Creación de entorno para aprender Ansible
  - 3 Máquinas virtuales en red interna:



Tutorial Ansible

VagrantFile

- Cr
  -

```
nodes = [
  { hostname: 'myCentosMachine',    ip: '192.168.100.2',    box: 'bento/centos-8.3',        ram: '2048' },
  { hostname: 'myUbuntuMachine',    ip: '192.168.100.3',    box: 'bento/ubuntu-20.04',      ram: '2048' },
  { hostname: 'myArchMachine',      ip: '192.168.100.4',    box: 'archlinux/archlinux',     ram: '2048' },
]

VAGRANTFILE_API_VERSION = 2
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
    nodes.each do |node|
        config.vm.define node[:hostname] do |nodeconfig|
            nodeconfig.vm.box = node[:box]
            nodeconfig.ssh.insert_key = false
            nodeconfig.vm.hostname = node[:hostname]
            nodeconfig.vm.network :private_network, ip: node[:ip]

            #Install multicast DNS to connect the guests by hostnames
            config.vm.provision "shell", inline: <<-SHELL
                DISTRO=$(cat /etc/os-release | head -1 | grep -o '".*"'  | tr -d '"')

                if [ "${DISTRO}" = "CentOS Linux" ]; then
                    yum install -y --quiet epel-release
                    yum install -y --quiet avahi nss-mdns
                    /bin/systemctl enable avahi-daemon
                    /bin/systemctl start avahi-daemon
                fi

                if [ "$DISTRO" = "Ubuntu" ]; then
                    apt-get install -y avahi-daemon avahi-utils libnss-mdns
                fi

                if [ "$DISTRO" = "Arch Linux" ]; then
                    pacman -S avahi nss-mdns --noconfirm
                    systemctl enable avahi-daemon
                    systemctl start avahi-daemon
                fi
            SHELL

            #Configure the resources of the virtual machine
            config.vm.provider "virtualbox" do |v|
                v.memory = node[:ram]
            end
        end
    end
end
```

Lista con características de las máquinas

Configuramos cada máquina

Ejecutamos comandos en cada máquina

Indicamos proveedor y recursos de la máquina

Tutorial Ansible

- Instalación de Ansible:
  - Instalar requisitos de Ansible en todos los nodos
    - **Crear usuario con permisos sudo**

    Ejemplo:

vagrant@myCentosMachine:~

```
[vagrant@myCentosMachine ~]$ sudo useradd --create-home --groups wheel -p $(open
ssl passwd -1 myP4SSw0rd) ansibleuser
```

vagrant@myUbuntuMachine: ~

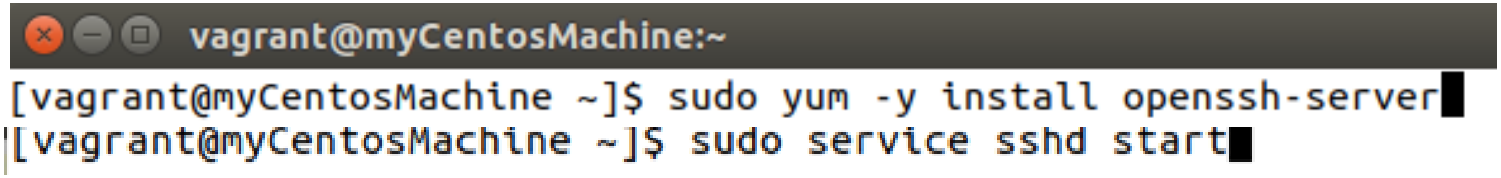
```
vagrant@myUbuntuMachine:~$ sudo useradd --create-home --group sudo -s /bin/bash
-s /bin/bash ansibleuser && echo ansibleuser:myP4SSw0rd | sudo chpasswd --crypt-
method=SHA512
```

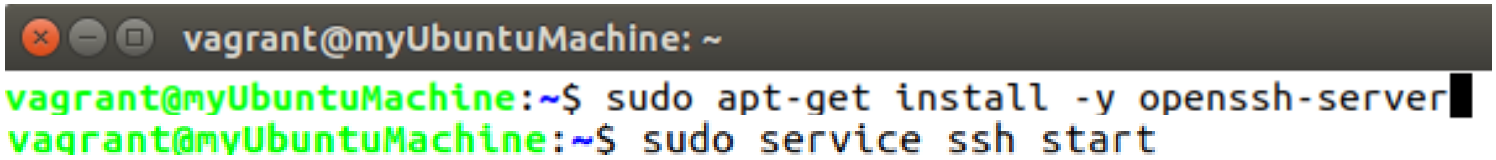
vagrant@myArchMachine:~

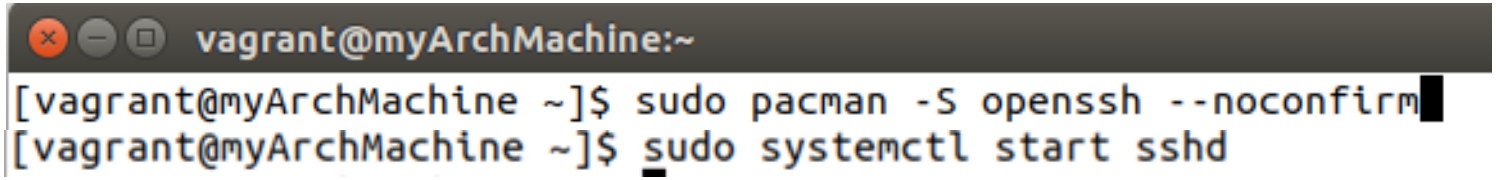
```
[vagrant@myArchMachine ~]$ sudo sed --in-place 's/^#\s*\(%wheel\s\+ALL=(ALL)\s\+ALL\)/\1/' /etc/sudoers
[vagrant@myArchMachine ~]$ sudo useradd --create-home --group wheel -s /bin/bash -s /bin/bash ansibleuser &&
 echo ansibleuser:myP4SSw0rd | sudo chpasswd --crypt-method=SHA512
[vagrant@myArchMachine ~]$
```

Tutorial Ansible

- Instalación de Ansible:
  - Instalar requisitos de Ansible en todos los nodos
    - Crear usuario con permisos sudo
    - **Instalar SSH**



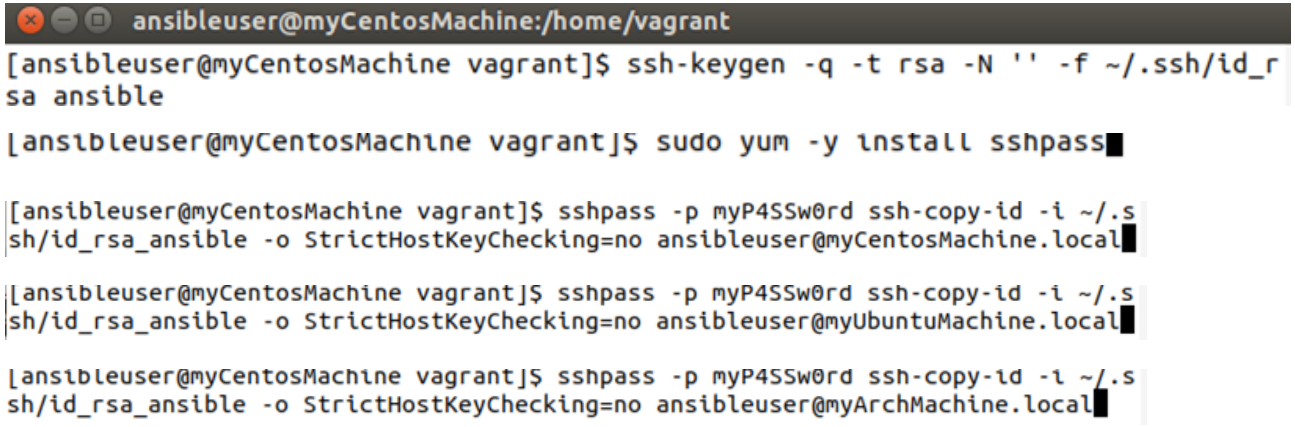




Tutorial Ansible

- Instalación de Ansible:
  - Instalar requisitos de Ansible en todos los nodos
    - Crear usuario con permisos sudo
    - Instalar SSH
    - **Configurar SSH para comunicación Control -> Managed**

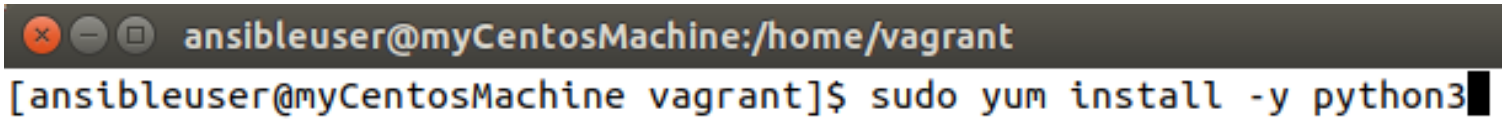
Ejemplo:



No es recomendable pasar la contraseña en plano en el comando. Es mejor introducirla cuando nos la pida

Tutorial Ansible

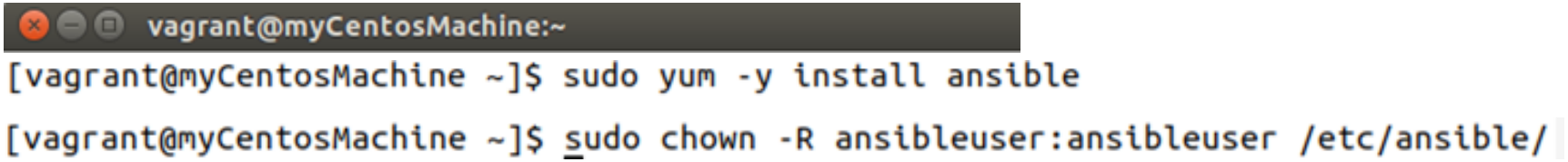
- Instalación de Ansible:
  - Instalar requisitos de Ansible en todos los nodos
    - Crear usuario con permisos sudo
    - Instalar SSH
    - Configurar SSH para comunicación Control -> Managed
    - **Instalar Python**



El resto de máquinas ya tienen instalado Python. Si no lo tuvieran, se tendría que instalar

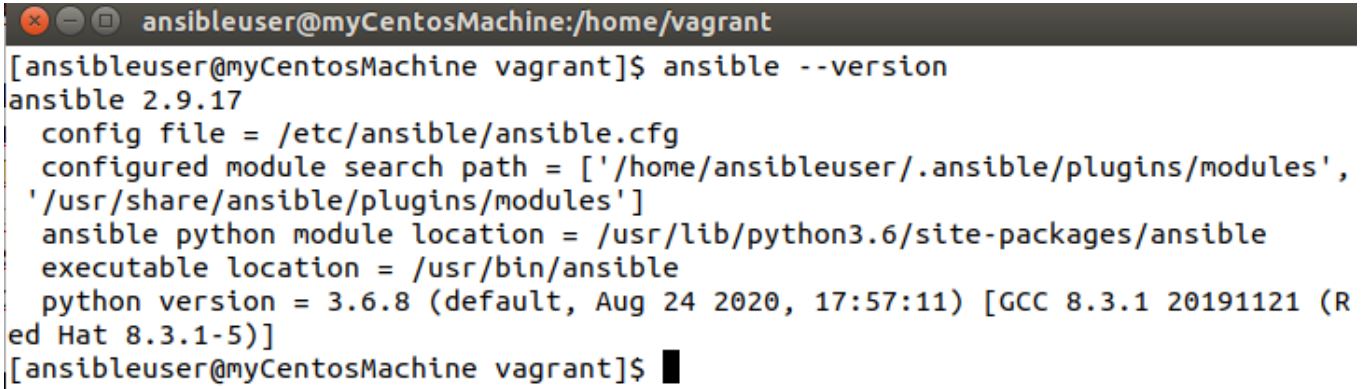
Tutorial Ansible

- Instalación de Ansible:
  - Instalar requisitos de Ansible en todos los nodos
    - Crear usuario con permisos sudo
    - Instalar SSH
    - Configurar SSH para comunicación Control -> Managed
    - Instalar Python
  - **Instalar Ansible en el Control Node**



Tutorial Ansible

- Instalación de Ansible:
  - Instalar requisitos de Ansible en todos los nodos
    - Crear usuario con permisos sudo
    - Instalar SSH
    - Configurar SSH para comunicación Control -> Managed
    - Instalar Python
  - Instalar Ansible en el Control Node
- **Comprobar que se instaló correctamente**



Tutorial Ansible

- Inventario
  - En comando
    - **Con IPs**

Módulo ping: nos devuelve Pong si tenemos acceso a la máquina remota

```
ansibleuser@myCentosMachine:/home/vagrant
[ansibleuser@myCentosMachine vagrant]$ ansible all --inventory="192.168.100.3,"
--private-key ~/.ssh/id_rsa_ansible -m ping
192.168.100.3 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
[ansibleuser@myCentosMachine vagrant]$
```

Tutorial Ansible

- Inventario
  - En comando
    - Con IPs
    - **Con nombres**

Módulo ping: nos devuelve Pong si tenemos acceso a la máquina remota

```
ansibleuser@myCentosMachine:/home/vagrant
[ansibleuser@myCentosMachine vagrant]$ ansible all --inventory="myUbuntuMachine.
local," --private-key ~/.ssh/id_rsa_ansible -m ping
myUbuntuMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
[ansibleuser@myCentosMachine vagrant]$
```

Tutorial Ansible

- Inventario
  - En comando
    - Con IPs
    - **Con nombres**

Se pueden indicar varios servidores

Módulo ping: nos devuelve Pong si tenemos acceso a la máquina remota

```
ansibleuser@myCentosMachine:/home/vagrant
[ansibleuser@myCentosMachine vagrant]$ ansible all --inventory="myUbuntuMachine.
local,myArchMachine.local,myCentosMachine.local" --private-key ~/.ssh/id_rsa_ans
ible -m ping
myUbuntuMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
[WARNING]: Platform linux on host myArchMachine.local is using the discovered
Python interpreter at /usr/bin/python, but future installation of another
Python interpreter could change this. See https://docs.ansible.com/ansible/2.9/
reference_appendices/interpreter_discovery.html for more information.
myArchMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
myCentosMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
[ansibleuser@myCentosMachine vagrant]$
```

Tutorial Ansible

- Inventario
  - En comando
    - Con IPs
    - Con nombres
  - En Archivo
    - **Por defecto**

Módulo ping: nos devuelve Pong si tenemos acceso a la máquina remota

```
ansibleuser@myCentosMachine:/home/vagrant
[ansibleuser@myCentosMachine vagrant]$ ansible all --private-key ~/.ssh/id_rsa_a
nsible -m ping
[WARNING]: Platform linux on host 192.168.100.4 is using the discovered Python
interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this. See https://docs.ansible.com/ansible/2.9/referen
ce_appendices/interpreter_discovery.html for more information.
192.168.100.4 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
192.168.100.3 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
192.168.100.2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
[ansibleuser@myCentosMachine vagrant]$
```

Inventario **con Ips (o nombres)**

```
ansibleuser@myCentosMachine:/home/vagrant
cat /etc/ansible/hosts
192.168.100.2
192.168.100.3
192.168.100.4
```

Formato: INI, YAML o JSON

Tutorial Ansible

- Inventario
  - En comando
    - Con IPs
    - Con nombres
  - En Archivo
    - **Por defecto**

Módulo ping: nos devuelve Pong si tenemos acceso a la máquina remota

```
ansibleuser@myCentosMachine:/home/vagrant
[ansibleuser@myCentosMachine vagrant]$ ansible myProductionEnvironment --private
-key ~/.ssh/id_rsa_ansible -m ping
myCentosMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
myUbuntuMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
[ansibleuser@myCentosMachine vagrant]$
```

Inventario **con entornos**

```
ansibleuser@myCentosMachine:/home/vagrant
cat /etc/ansible/hosts
[myTestEnvironment]
myArchMachine.local

[myProductionEnvironment]
myCentosMachine.local
myUbuntuMachine.local
```

Tutorial Ansible

- Inventario
  - En comando
    - Con IPs
    - Con nombres
  - En Archivo
    - **Por defecto**

Módulo ping: nos devuelve Pong si tenemos acceso a la máquina remota

Se configuran dos entornos

```
ansibleuser@myCentosMachine:/home/vagrant
[ansibleuser@myCentosMachine vagrant]$ ansible myProductionEnvironment:myTestEnv
ironment --private-key ~/.ssh/id_rsa_ansible -m ping
myUbuntuMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
[WARNING]: Platform linux on host myArchMachine.local is using the discovered
Python interpreter at /usr/bin/python, but future installation of another
Python interpreter could change this. See https://docs.ansible.com/ansible/2.9/
reference_appendices/interpreter_discovery.html for more information.
myArchMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
myCentosMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
[ansibleuser@myCentosMachine vagrant]$
```

Inventario **con entornos**

```
ansibleuser@myCentosMachine:/home/vagrant
cat /etc/ansible/hosts
[myTestEnvironment]
myArchMachine.local

[myProductionEnvironment]
myCentosMachine.local
myUbuntuMachine.local
```

Tutorial Ansible

- Inventario
  - En comando
    - Con IPs
    - Con nombres
  - En Archivo
    - **Por defecto**

Módulo ping: nos devuelve Pong si tenemos acceso a la máquina remota

Se configuran todos los entornos

```
ansibleuser@myCentosMachine:/home/vagrant
[ansibleuser@myCentosMachine vagrant]$ ansible all --private-key ~/.ssh/id_rsa_a
nsible -m ping
[WARNING]: Platform linux on host myArchMachine.local is using the discovered
Python interpreter at /usr/bin/python, but future installation of another
Python interpreter could change this. See https://docs.ansible.com/ansible/2.9/
reference_appendices/interpreter_discovery.html for more information.
myArchMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
myUbuntuMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
myCentosMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
[ansibleuser@myCentosMachine vagrant]$
```

Inventario **con entornos**

```
ansibleuser@myCentosMachine:/home/vagrant
cat /etc/ansible/hosts
[myTestEnvironment]
myArchMachine.local

[myProductionEnvironment]
myCentosMachine.local
myUbuntuMachine.local
```

Tutorial Ansible

- Inventario
  - En comando
    - Con IPs
    - Con nombres
  - En Archivo
    - **Por defecto**

Módulo ping: nos devuelve Pong si tenemos acceso a la máquina remota

Se configuran las máquinas sin entorno

```
ansibleuser@myCentosMachine:/home/vagrant
[ansibleuser@myCentosMachine vagrant]$ ansible ungrouped --private-key ~/.ssh/id
_rsa_ansible -m ping
myCentosMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
[ansibleuser@myCentosMachine vagrant]$
```

Inventario **con entornos**

```
ansibleuser@myCentosMachine:/home/vagrant
cat /etc/ansible/hosts
myCentosMachine.local

[myTestEnvironment]
myArchMachine.local

[myProductionEnvironment]
myUbuntuMachine.local
```

Hay un entorno por defecto para las máquinas sin asignar a un entorno: ungrouped

Tutorial Ansible

- Inventario
  - En comando
    - Con IPs
    - Con nombres
  - En Archivo
    - **Por defecto**

Módulo ping: nos devuelve Pong si tenemos acceso a la máquina remota

```
ansibleuser@myCentosMachine:/home/vagrant
[ansibleuser@myCentosMachine vagrant]$ ansible myProductionEnvironment:Asturias
--private-key ~/.ssh/id_rsa_ansible -m ping
myUbuntuMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
[ansibleuser@myCentosMachine vagrant]$
```

Sólo se configura una vez

Inventario **con entornos**

```
ansibleuser@myCentosMachine:/home/vagrant
cat /etc/ansible/hosts
myCentosMachine.local

[myTestEnvironment]
myArchMachine.local

[myProductionEnvironment]
myUbuntuMachine.local

[Asturias]
myUbuntuMachine.local
```

Una misma máquina puede estar en dos entornos ej. myUbuntuMachine

myProductionEnvironment:
- DBProductionServers:
  * myUbuntuMachine
  * myArchMachine
- webProductionServer:
  * myCentosMachine

Tutorial Ansible

- Inventario
  - En comando
    - Con IPs
    - Con nombres
  - En Archivo
    - **Por defecto**

Módulo ping: nos devuelve Pong si tenemos acceso a la máquina remota

Se puede ejecutar un sub-entorno

```
ansibleuser@myCentosMachine:/home/vagrant
[ansibleuser@myCentosMachine vagrant]$ ansible DBProductionServers --private-key
 ~/.ssh/id_rsa_ansible -m ping
myUbuntuMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
[WARNING]: Platform linux on host myArchMachine.local is using the discovered
Python interpreter at /usr/bin/python, but future installation of another
Python interpreter could change this. See https://docs.ansible.com/ansible/2.9/
reference_appendices/interpreter_discovery.html for more information.
myArchMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
[ansibleuser@myCentosMachine vagrant]$
```

Inventario **con sub-entornos**

```
ansibleuser@myCentosMachine:/home/vagrant
cat /etc/ansible/hosts
[myProductionEnvironment:children]
DBProductionServers
WebProductionServer

[DBProductionServers]
myUbuntuMachine.local
myArchMachine.local

[WebProductionServer]
myCentosMachine.local
```

myProductionEnvironment:
- DBProductionServers:
  * myUbuntuMachine
  * myArchMachine
- webProductionServer:
  * myCentosMachine

Tutorial Ansible

- Inventario
  - En comando
    - Con IPs
    - Con nombres
  - En Archivo
    - **Por defecto**

Módulo ping: nos devuelve Pong si tenemos acceso a la máquina remota

Se puede ejecutar el entorno

```
ansibleuser@myCentosMachine:/home/vagrant
[ansibleuser@myCentosMachine vagrant]$ ansible myProductionEnvironment -private
-key ~/.ssh/id_rsa_ansible -m ping
[WARNING]: Platform linux on host myArchMachine.local is using the discovered
Python interpreter at /usr/bin/python, but future installation of another
Python interpreter could change this. See https://docs.ansible.com/ansible/2.9/
reference_appendices/interpreter_discovery.html for more information.
myArchMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
myUbuntuMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
myCentosMachine.local | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
[ansibleuser@myCentosMachine vagrant]$
```

Inventario **con sub-entornos**

```
ansibleuser@myCentosMachine:/home/vagrant
cat /etc/ansible/hosts
[myProductionEnvironment:children]
DBProductionServers
WebProductionServer

[DBProductionServers]
myUbuntuMachine.local
myArchMachine.local

[WebProductionServer]
myCentosMachine.local
```

Tutorial Ansible

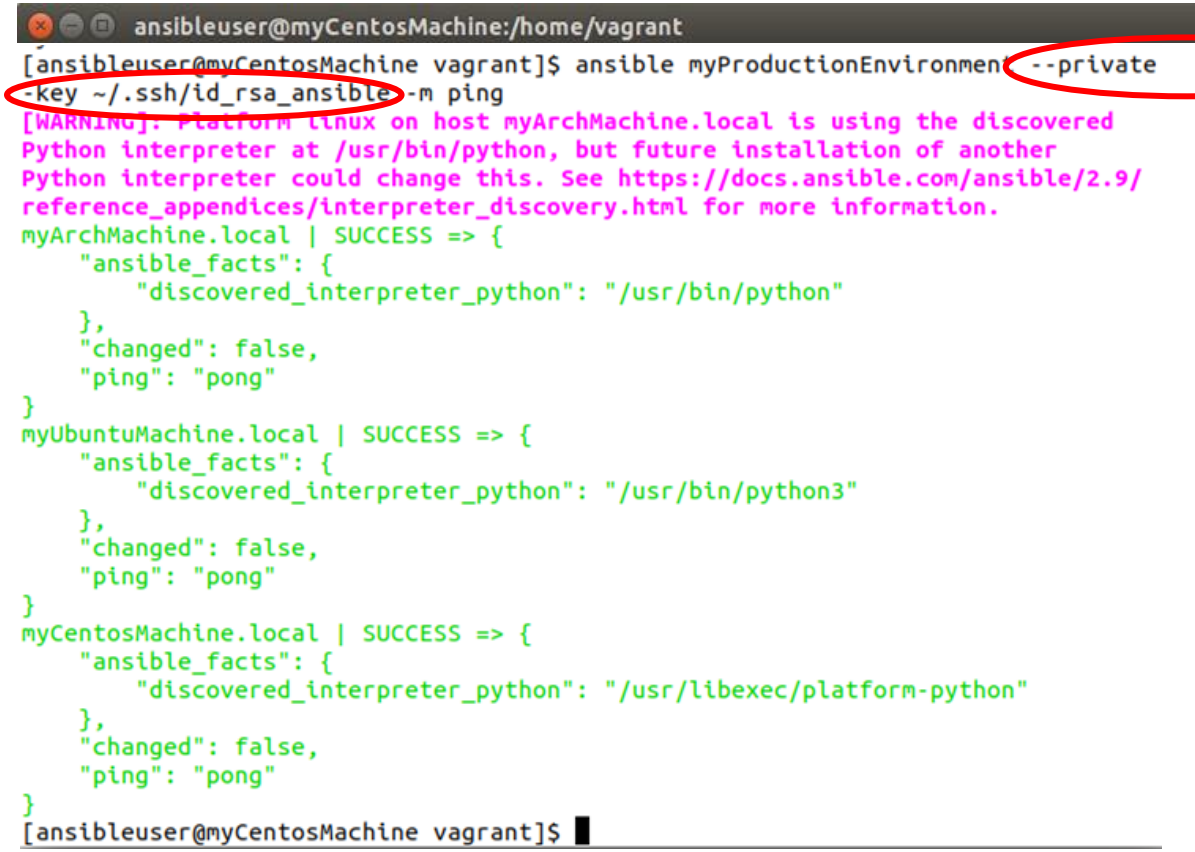
- Inventario
  - En comando
    - Con IPs
    - Con nombres
  - En Archivo
    - Por defecto
    - **Personalizado**
      - Creamos un archivo con el inventario
      - Se tiene que indicar el PATH del inventario
        - `ansible --inventory=PATH`
        - `ansible --inventory=PATH1 --inventory=PATH2`
  - Dinámico: https://docs.ansible.com/ansible/latest/user_guide/intro_dynamic_inventory.html

Tutorial Ansible

- Inventario: variables
  - **Comando**

Módulo ping: nos devuelve Pong si tenemos acceso a la máquina remota

--private-key: clave privada que utilizará el Control Node para conectarse a cada máquina



Tutorial Ansible

Módulo ping: nos devuelve Pong si tenemos acceso a la máquina remota

```
ansibleuser@myCentosMachine:/home/vagrant
[ansibleuser@myCentosMachine ~]$ ansible all -m ping
myCentosMachine.local | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
myArchMachine.local | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
myUbuntuMachine.local | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[ansibleuser@myCentosMachine ~]$
```

- Inventario: variables
  - Comando
  - **Inventario**

Inventario **con parámetros**

```
ansibleuser@myCentosMachine:/home/vagrant
cat /etc/ansible/hosts

[myTestEnvironment]
myArchMachine.local ansible_python_interpreter=/usr/bin/python

[myProductionEnvironment]
myCentosMachine.local ansible_connection=local
myUbuntuMachine.local

[myProductionEnvironment:vars]
ansible_python_interpreter=/usr/bin/python3

[all:vars]
ansible_ssh_private_key_file=~/.ssh/id_rsa_ansible
```

Python que utilizará Ansible en esa máquina

Sólo afecta a una máquina

Cómo se conecta Ansible a esa máquina para configurarla ej. ssh, local, etc

Afecta a las máquinas de un entorno

Afecta a todas las máquinas

Tutorial Ansible

- Inventario: variables
  - Comando
  - Inventario
  - **Carpeta /etc/ansible/**

Las variables de una máquina se guardan en carpeta host_vars + nombre de la máquina (puede ser yml json o ini)

```
ansibleuser@myCentosMachine:~
cat /etc/ansible/host_vars/myArchMachine.local.yml
ansible_python_interpreter: /usr/bin/python
```

```
ansibleuser@myCentosMachine:~
cat /etc/ansible/host_vars/myCentosMachine.local
ansible connection: local
```

Las de un entorno en la carpeta group_vars + nombre del entorno

```
ansibleuser@myCentosMachine:~
cat /etc/ansible/group_vars/myProductionEnvironment.yml
ansible_python_interpreter: /usr/bin/python3
```

Las de todas las máquinas en group_vars + all

```
ansibleuser@myCentosMachine:~
cat /etc/ansible/group_vars/all.yml
ansible_ssh_private_key_file: ~/.ssh/id_rsa
ansible_sudo_pass: myP4SSw0rd
```

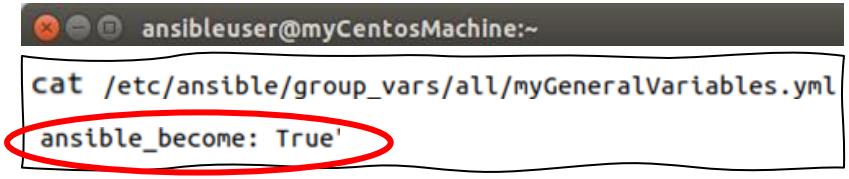
Inventario

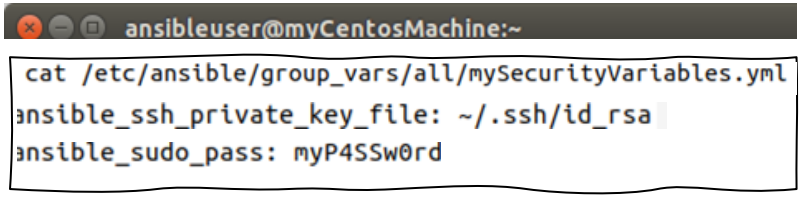
```
ansibleuser@myCentosMachine:/home/vagrant
cat /etc/ansible/hosts

[myTestEnvironment]
myArchMachine.local

[myProductionEnvironment]
myCentosMachine.local
myUbuntuMachine.local
```

+

Tutorial Ansible

- Inventario: variables
  - Comando
  - Inventario
  - **Carpeta /etc/ansible/**
    - Se pueden crear subcarpetas





Permite configurar máquinas remotas con permisos sudo -> No es recomendable
Es mejor indicarlo cuando se necesite

Tutorial Ansible

- Inventario: variables
  - Comando
  - Inventario
  - Carpeta /etc/ansible/
    - Se pueden crear subcarpetas
  - **Se pueden crear variables existentes o personalizadas para luego utilizarlas en la configuración**
    - Ej. `myUbuntuMachine.local localBackup=/etc/backups/db proxy=databaseProxy.example.com`

Tutorial Ansible

- Inventario: variables encriptadas
  - Ansible Vault:
    - **encrypt**

Indicamos la clave de encriptado/desencriptado

```
ansibleuser@myCentosMachine:~
ansible-vault encrypt /etc/ansible/group_vars/all/mySecurityVariables.yml
New Vault password:
Confirm New Vault password:
Encryption successful
```

```
ansibleuser@myCentosMachine:~
cat /etc/ansible/group_vars/all/mySecurityVariables.yml
ansible_sudo_pass: myP4SSw0rd
```

```
ansibleuser@myCentosMachine:~
cat /etc/ansible/group_vars/all/mySecurityVariables.yml
$ANSIBLE_VAULT;1.1;AES256
30366436376430323165326334343461343664373132663133623435366330663030383333363336
30613830343166636133376234323034353334396431613 70a3632373466626139373136316566 63
35663464313563323661633831373632656365396434313833303961393536353864646438353861
62336533376365636 40a37346161333136383136376530386161323365643066386461613765663 8
39653336653537333039383334383335623135333363326262623964366665363935
```

Tutorial Ansible

- Inventario: variables encriptadas
  - Ansible Vault:
    - **encrypt**

Falla si no indicamos la clave de desencriptado

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible all -m ping
ERROR! Attempting to decrypt but no vault secrets found
```

Alternativa 1: que el usuario la introduzca cuando se haga la ejecución

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible all -m ping --ask-vault-pass
Vault password:
myCentosMachine.local | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
myArchMachine.local | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
myUbuntuMachine.local | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[ansibleuser@myCentosMachine ~]$
```

Tutorial Ansible

- Inventario: variables encriptadas
  - Ansible Vault:
    - **encrypt**

Falla si no indicamos la clave de desencriptado

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible all -m ping
ERROR! Attempting to decrypt but no vault secrets found
```

Alternativa 2: guardarla en un archivo

```
ansibleuser@myCentosMachine:~
echo "myVaultPass" > ~/.my_ansible_vault_password
```

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible all -m ping --vault-password-file ~/.my_ansible_vault
_password
myCentosMachine.local | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
myUbuntuMachine.local | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
myArchMachine.local | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[ansibleuser@myCentosMachine ~]$
```

Tutorial Ansible

- Inventario: variables encriptadas
  - Ansible Vault:
    - **encrypt**

Falla si no indicamos la clave de desencriptado

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible all -m ping
ERROR! Attempting to decrypt but no vault secrets found
```

Alternativa 3: Modificar la configuración de Ansible

```
ansibleuser@myCentosMachine:~
echo "myVaultPass" > ~/.my_ansible_vault_password
```

Editamos el archivo /etc/ansible/ansible.cfg
Añadimos: vault_password_file = ~/.my_ansible_vault_password

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ sed --in-place 's/#vault_password_file = \/path\/to\/vault_pa
ssword_file/vault_password_file = ~\/.my_ansible_vault_password/' /etc/ansible/ansible.cfg
```

Utiliza automáticamente la contraseña de desencriptación (y las variables encriptadas)

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible all -m ping
myCentosMachine.local | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
myUbuntuMachine.local | SUCCESS => {
```

Tutorial Ansible

- Inventario: variables encriptadas
  - Ansible Vault:
    - encrypt
    - view: ver un archivo encriptado (seguirá encriptado)
    - edit: editar un archivo encriptado (seguirá encriptado)
    - decrypt
    - create: crea un archivo y lo guarda encriptado
    - rekey: cambia la clave de encriptamiento
    - Se pueden tener múltiples contraseñas de encriptado -> IDs Vault
    - encrypt-string: encripta una única variable
    - La clave de (des)encriptado se puede guardar en un secret manager o keyring -> --vault-password-file algo-client.py
      - algo-client.py es un programa que tenemos que crear para proporcionar la clave en la salida estándar

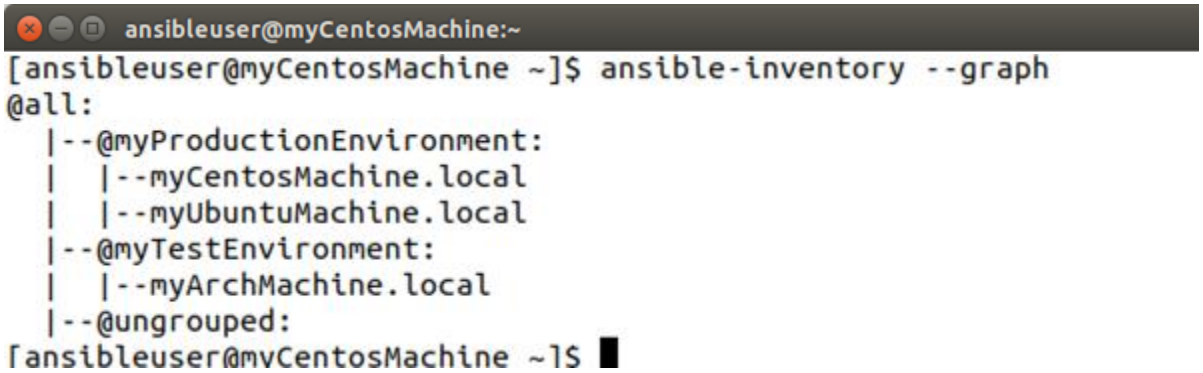
Tutorial Ansible

- Inventario: imprimir entornos, máquinas y variables

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible-inventory --list
{
    "_meta": {
        "hostvars": {
            "myArchMachine.local": {
                "ansible_become": true,
                "ansible_python_interpreter": "/usr/bin/python",
                "ansible_ssh_private_key_file": "~/.ssh/id_rsa_ansible",
                "ansible_sudo_pass": "myP4SSw0rd"
            },
            "myCentosMachine.local": {
                "ansible_become": true,
                "ansible_connection": "local",
                "ansible_python_interpreter": "/usr/bin/python3",
                "ansible_ssh_private_key_file": "~/.ssh/id_rsa_ansible",
                "ansible_sudo_pass": "myP4SSw0rd"
            },
            "myUbuntuMachine.local": {
                "ansible_become": true,
                "ansible_python_interpreter": "/usr/bin/python3",
                "ansible_ssh_private_key_file": "~/.ssh/id_rsa_ansible",
                "ansible_sudo_pass": "myP4SSw0rd"
            }
        }
    },
    "all": {
        "children": [
            "myProductionEnvironment",
            "myTestEnvironment",
            "ungrouped"
        ]
    },
    "myProductionEnvironment": {
        "hosts": [
            "myCentosMachine.local",
            "myUbuntuMachine.local"
        ]
    },
    "myTestEnvironment": {
        "hosts": [
            "myArchMachine.local"
        ]
    }
}
```

Tutorial Ansible

- Inventario: imprimir entornos y máquinas



Tutorial Ansible

- Inventario: imprimir variables

Tutorial Ansible

- Inventario: imprimir una variable

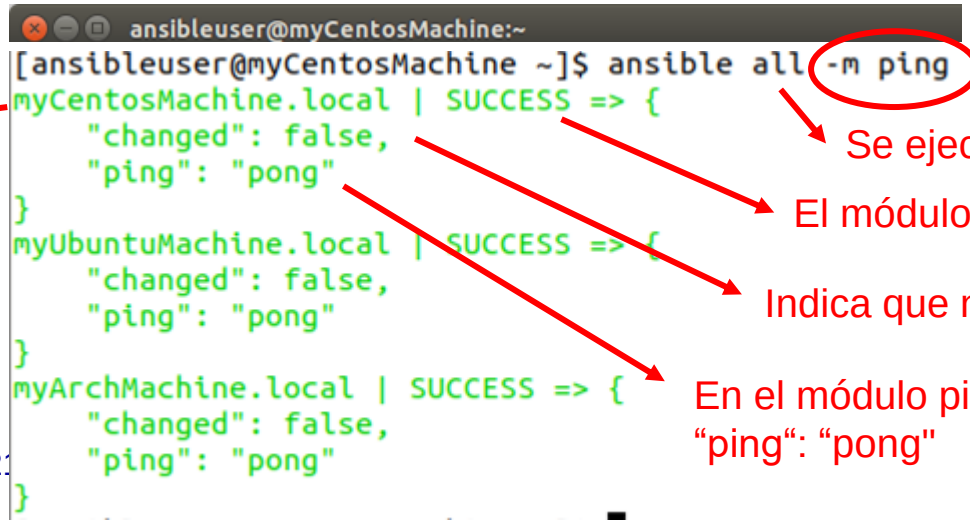
```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible-inventory --host myCentosMachine.local ;
ansible-inventory --host myUbuntuMachine.local ; ansible-inventory --host myArchMa
chine.local
{
    "ansible_connection": "local",
    "ansible_python_interpreter": "/usr/bin/python3"
}
{
    "ansible_python_interpreter": "/usr/bin/python3"
}
{
    "ansible_python_interpreter": "/usr/bin/python"
}
```

Tutorial Ansible

- Módulos
  - Indican qué es lo queremos tener (o no tener) en la infraestructura
    - Necesito tener un paquete instalado
    - Necesito tener un servicio arrancado
    - …
  - Hay varios módulos creados
  - Modos de ejecución:
    - Comando
    - Playbook: tendremos varias tareas donde cada una es un módulo
    - Ansible Tower: interfaz visual para ejecutar Ansible

Tutorial Ansible

- Módulos
  - Comando: `ansible patron_de_maquinas -m nombre_del_modulo -a "[argumentos_del_modulo]“`
  - Módulo ping: Comprueba que se puede utilizar Ansible en la máquina remota
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/ping_module.html



Máquina sobre la que se ejecuta

Módulo ping

Se ejecuta el módulo en todas las máquinas

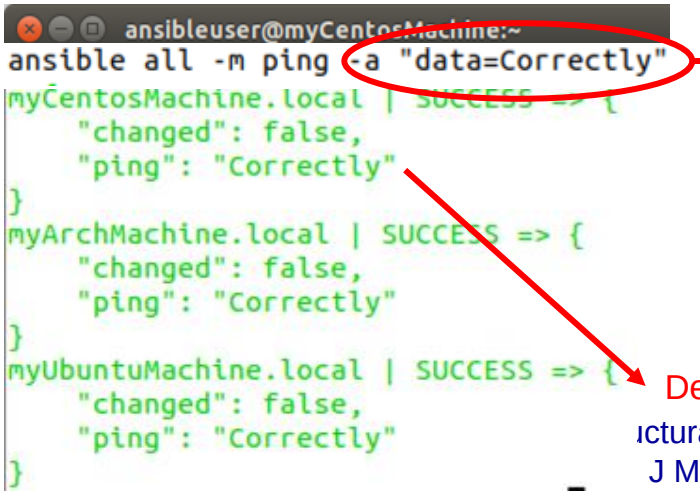
El módulo se ejecuta correctamente

Indica que no cambió el estado de la máquina

En el módulo ping si todo fue correcto, devuelve “ping“: “pong"

Tutorial Ansible

- Módulos
  - Comando: `ansible patron_de_maquinas -m nombre_del_modulo -a "[argumentos_del_modulo]“`
  - Módulo ping: Comprueba que se puede utilizar Ansible en la máquina remota
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/ping_module.html



Parámetros con sus valores:

Parameters

Parameter	Choices/Defaults	Comments
data string	Default: "pong"	Data to return for the `ping` return value. If this parameter is set to `crash`, the module will cause an exception.

Devuelve Correctly porque cambiamos el parámetro “data”

ıctura como código

Tutorial Ansible

- Módulos
  - Módulo user: Gestiona usuarios y permisos
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/user_module.html

Algunos parámetros son obligatorios

name string / required		Name of the user to create, remove or modify. aliases: user

Otros parámetros vienen con opciones por defecto

create_home boolean	**Choices:** • no • **yes** ←	Unless set to `no`, a home directory will be made for the user when the account is created or if the home directory does not exist. Changed from `createhome` to `create_home` in Ansible 2.5. aliases: createhome

Tutorial Ansible

- Módulos
  - Módulo user: Gestiona usuarios y permisos
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/user_module.html

Quiero tener en todas las máquinas un usuario my_user1 sin carpeta home

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible all -m user -a "name=my_user1 create_home=False"
myCentosMachine.local | FAILED! => {
    "changed": false,
    "msg": "useradd: Permission denied.\nuseradd: cannot lock /etc/passwd; try again later.\n",
    "name": "my_user1",
    "rc": 1
}
myUbuntuMachine.local | FAILED! => {
    "changed": false,
    "msg": "useradd: Permission denied.\nuseradd: cannot lock /etc/passwd; try again later.\n",
    "name": "my_user1",
    "rc": 1
}
myArchMachine.local | FAILED! => {
    "changed": false,
    "msg": "useradd: Permission denied.\nuseradd: cannot lock /etc/passwd; try again later.\n",
    "name": "my_user1",
    "rc": 1
}
[ansibleuser@myCentosMachine ~]$
```

Se utilizan varios parámetros

Este módulo requiere permisos sudo en cada máquina

Tutorial Ansible

- Módulos
  - Módulo user: Gestiona usuarios y permisos
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/user_module.html

Quiero tener en todas las máquinas un usuario my_user1 sin carpeta home

Fallan si no indicamos en el inventario la contraseña sudo de cada máquina

Permite ejecutar comandos como sudo

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible all -m user -a "name=my_user1 create_home=False" --become
myCentosMachine.local | FAILED! => {
    "changed": false,
    "module_stderr": "sudo: a password is required\n",
    "module_stdout": "",
    "msg": "MODULE FAILURE\nSee stdout/stderr for the exact error",
    "rc": 1
}
myArchMachine.local | FAILED! => {
    "msg": "Missing sudo password"
}
```

Tutorial Ansible

- Módulos
  - Módulo user: Gestiona usuarios y permisos
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/user_module.html

Quiero tener en todas las máquinas un usuario my_user1 sin carpeta home

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible all -m user -a "name=my_user1 create_home=F
alse" --become --ask-become-pass
BECOME password:
myArchMachine.local | CHANGED => {
    "changed": true,
    "comment": "",
    "create_home": false,
    "group": 1002,
    "home": "/home/my_user1",
    "name": "my_user1",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 1002
}
myCentosMachine.local | CHANGED => {
    "changed": true,
    "comment": "",
    "create_home": false,
    "group": 1002,
    "home": "/home/my_user1",
    "name": "my_user1",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 1002
}
```

Permite ejecutar comandos como sudo y le indicamos que vamos a proporcionar la contraseña sudo

Introducimos la contraseña

Si la máquina no tiene el usuario como lo queremos, lo crea/modifica

Si la máquina ya tiene el usuario como lo queremos, no hace nada

```
myUbuntuMachine.local | SUCCESS => {
    "append": false,
    "changed": false,
    "comment": "",
    "group": 1002,
    "home": "/home/my_user1",
    "move_home": false,
    "name": "my_user1",
    "shell": "/bin/sh",
    "state": "present",
    "uid": 1002
}
```

Tutorial Ansible

- Módulos
  - Módulo user: Gestiona usuarios y permisos
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/user_module.html

Quiero tener en todas las máquinas un usuario my_user2 sin carpeta home

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible all -m user -a "name=my_user2 create_home=F
alse" --become --extra-vars 'ansible_become_pass=myP4SSw0rd'
myArchMachine.local | CHANGED => {
    "changed": true,
    "comment": "",
    "create_home": false,
    "group": 1003,
    "home": "/home/my_user2",
    "name": "my_user2",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 1003
}
myCentosMachine.local | CHANGED => {
    "changed": true,
    "comment": "",
    "create_home": false,
    "group": 1003,
    "home": "/home/my_user2",
    "name": "my_user2"
```

Podemos indicar la contraseña sudo en el propio comando

Tutorial Ansible

- Módulos
  - Módulo user: Gestiona usuarios y permisos
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/user_module.html

Quiero tener en todas las máquinas un usuario my_user3 sin carpeta home

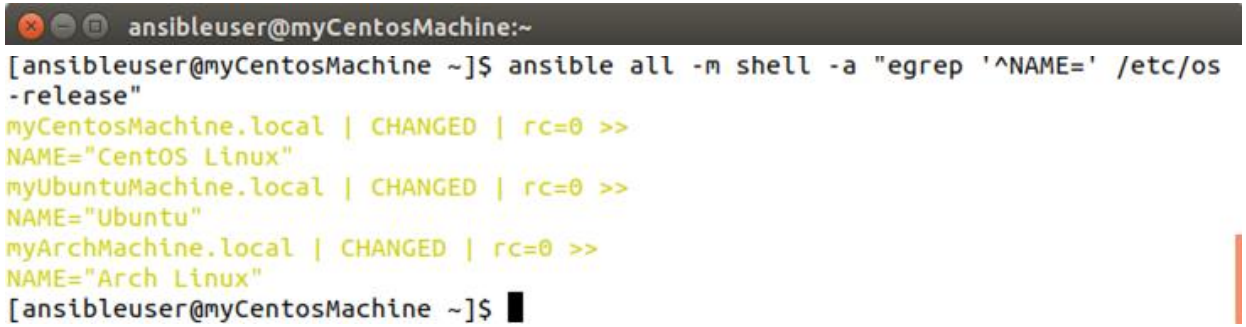
```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible all -m user -a "name=my_user3 create_home=F
alse"
myUbuntuMachine.local | SUCCESS => {
    "append": false,
    "changed": false,
    "comment": "",
    "group": 1004,
    "home": "/home/my_user3",
    "move_home": false,
    "name": "my_user3",
    "shell": "/bin/sh",
    "state": "present",
    "uid": 1004
}
myCentosMachine.local | SUCCESS => {
    "append": false,
    "changed": false,
    "comment": "",
    "group": 1004,
    "home": "/home/my_user3",
    "move_home": false,
```

Podemos guardar la contraseña en el inventario, Vault, etc.

Tutorial Ansible

- Módulos
  - Módulo shell: Ejecuta un comando en las máquinas
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/shell_module.html

Quiero ver la versión del sistema operativo Linux de cada máquina



Tutorial Ansible

- Módulos
  - Módulo package: gestor de paquetes genérico
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/package_module.html

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible all -m package -a "name=git" --become
--ask-become-pass
BECOME password:
myUbuntuMachine.local | SUCCESS => {
    "cache_update_time": 1608709951,
    "cache_updated": false,
    "changed": false
}
myArchMachine.local | CHANGED => {
    "changed": true,
    "msg": "installed 1 package(s). "
}
myCentosMachine.local | CHANGED => {
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
        "Installed: perl-Error-1:0.17025-2.el8.noarch",
        "Installed: perl-Git-2.27.0-1.el8.noarch",
        "Installed: emacs-filesystem-1:26.1-5.el8.noarch",
        "Installed: git-2.27.0-1.el8.x86_64",
        "Installed: git-core-2.27.0-1.el8.x86_64",
        "Installed: git-core-doc-2.27.0-1.el8.noarch"
    ]
}
[ansibleuser@myCentosMachine ~]$
```

No cambia porque ya tenía Git instalado

Cambia porque tuvo que instalar 1 paquete

Cambia porque tuvo que instalar varios paquetes

Quiero que todas mis máquinas tengan instalado git

Unas máquinas utilizarán yum, otras apt, etc.

IMPORTANTE: no todos los paquetes tienen el mismo nombre en las diferentes distribuciones ej. servidor web Apache en yum es "httpd", pero en apt es "apache2“ -> fallará en alguna máquina

Ansible también tiene módulos para gestores específicos

Tras ejecutar el módulo: todas las máquinas tienen instalado git

Tutorial Ansible

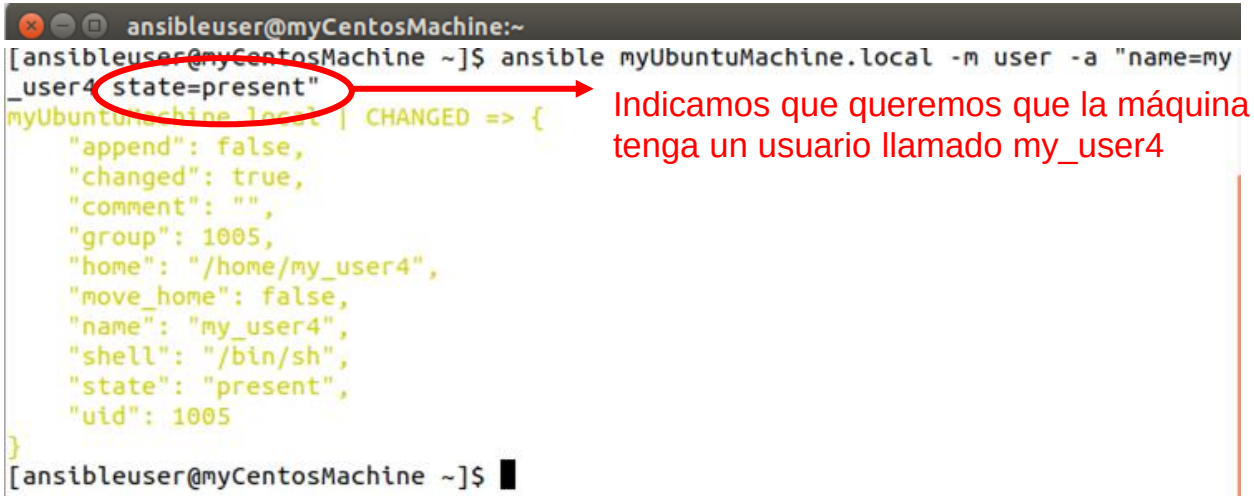
- Módulos: Estado
  - Son declarativos: se indica el estado al que se quiere llegar
  - Por ejemplo: `ansible all -m user -a "name=my_user4"`
    - No significa que se tiene que crear un usuario my_user4
    - Significado: Quiero que se tenga un usuario my_user4
      - Ansible no hace nada si ya existe el usuario
      - Ansible crea el usuario si no existe
  - Los módulos pueden tener diferentes estados:
    - Modulo user: state = present -> queremos tener ese usuario
    - Modulo user: state = absent -> queremos no tener ese usuario

Tutorial Ansible

- Módulos: Estado

Parámetro state (por defecto es present)

state string	**Choices:** • absent • **present** ←	Whether the account should exist or not, taking action if the state is different from what is stated.



Indicamos que queremos que la máquina myUbuntuMachine tenga un usuario llamado my_user4

Tutorial Ansible

- Módulos: Estado

Parámetro state (por defecto es present)

state string	**Choices:** • absent • **present** ←	Whether the account should exist or not, taking action if the state is different from what is stated.

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible all -m user -a "name=my_user4 state=absent"
myCentosMachine.local | SUCCESS => {
    "changed": false,
    "name": "my_user4",
    "state": "absent"
}
myUbuntuMachine.local | CHANGED => {
    "changed": true,
    "force": false,
    "name": "my_user4",
    "remove": false,
    "state": "absent"
}
myArchMachine.local | SUCCESS => {
    "changed": false,
    "name": "my_user4",
    "state": "absent"
}
[ansibleuser@myCentosMachine ~]$
```

Indicamos que queremos que ninguna máquina tenga un usuario llamado my_user4

Tenía un usuario my_user4, pero Ansible lo elimina

Ansible no hace nada porque la máquina ya no tenía el usuario my_user4

Tutorial Ansible

- Playbooks
  - Permiten ejecutar varios módulos (tareas)
  - Creamos un archivo .yml (o json o ini)

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myCreateUser.yml
---

- name: This is my first ansible playbook
  hosts: myProductionEnvironment
  become: True
  tasks:
  - name: My first task in the playbook creates an user
    user:
     name: myUserPlayBook1
     create_home: True
     state: present

[ansibleuser@myCentosMachine ~]$
```

Entorno del inventario a configurar

La configuración de todas las tareas se hará con permisos sudo en las máquinas remotas

Modulo a ejecutar, equivale a: `ansible myProductionEnvironment-m user -a "name=myUserPlayBook1 create_home=True state=present"`

Tutorial Ansible

- Playbooks
  - Permiten ejecutar varios módulos (tareas)
  - Creamos un archivo .yml (o json o ini)

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myCreateUser.yml
---

- name: This is my first ansible playbook
  hosts: myProductionEnvironment
  become: True
  tasks:
  - name: My first task in the playbook creates an user
    user:
     name: myUserPlayBook1
     create_home: True
     state: present

[ansibleuser@myCentosMachine ~]$
```

Comprobar si la sintaxis del Playbook es correcta

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible-playbook myCreateUser.yml --syntax-check

playbook: myCreateUser.yml
[ansibleuser@myCentosMachine ~]$
```

Tutorial Ansible

- Playbooks
  - Permiten ejecutar varios módulos (tareas)
  - Creamos un archivo .yml (o json o ini)

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myCreateUser.yml
---

- name: This is my first ansible playbook
  hosts: myProductionEnvironment
  become: True
  tasks:
  - name: My first task in the playbook creates an user
    user:
      name: myUserPlayBook1
      create_home: True
      state: present

[ansibleuser@myCentosMachir
```

Comprobar qué pasaría si ejecutásemos el Playbook

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible-playbook myCreateUser.yml --check

PLAY [This is my first ansible playbook] ***************************************

TASK [Gathering Facts] *********************************************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [My first task in the playbook creates an user] ***************************
changed: [myCentosMachine.local]
changed: [myUbuntuMachine.local]

PLAY RECAP *********************************************************************
myCentosMachine.local      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
myUbuntuMachine.local      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

No todos los módulos soportan check

Se ejecuta una tarea en 2 máquinas

Ambas máquinas cambiarán

Tutorial Ansible

- Playbooks
  - Permiten ejecutar varios módulos (tareas)
  - Creamos un archivo .yml (o json o ini)

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myCreateUser.yml
---

- name: This is my first ansible playbook
  hosts: myProductionEnvironment
  become: True
  tasks:
  - name: My first task in the playbook creates an user
    user:
     name: myUserPlayBook1
     create_home: True
     state: present

[ansibleuser@myCentosMachir
```

Ejecutar el playbook

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible-playbook myCreateUser.yml

PLAY [This is my first ansible playbook] ***************************************

TASK [Gathering Facts] *********************************************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [My first task in the playbook creates an user] ***************************
changed: [myUbuntuMachine.local]
changed: [myCentosMachine.local]

PLAY RECAP *********************************************************************
myCentosMachine.local      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
myUbuntuMachine.local      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Tutorial Ansible

- Playbooks

Se ejecutan
2 módulos

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myCreateUser.yml
---

- name: This is my first ansible playbook
  hosts: myProductionEnvironment
  become: True
  tasks:
  - name: My first task in the playbook creates an user
    user:
     name: myUserPlayBook1
     create_home: True
     state: present

  - name: My second task in the plabook creates a file in the user folder
    copy:
     content: "You must not share your password\n"
     dest: "/home/myUserPlayBook1/README.txt"

[ansibleuser@myCentosMachine ~]$
```

Primera tarea: no hace nada porque ya existen los usuarios

Segunda tarea: cambian las máquinas porque se les copia un README.txt

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible-playbook myCreateUser.yml

PLAY [This is my first ansible playbook] ***************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [My first task in the playbook creates an user] ***************************************************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [My second task in the plabook creates a file in the user folder] *********************************************
changed: [myCentosMachine.local]
changed: [myUbuntuMachine.local]

PLAY RECAP *********************************************************************************************************
myCentosMachine.local      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
myUbuntuMachine.local      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansibleuser@myCentosMachine ~]$
```

Tutorial Ansible

- Playbooks: iteraciones

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myCreateUser.yml
---

- name: This is my first ansible playbook
  hosts: myProductionEnvironment
  become: True
  tasks:
  - name: My first task in the playbook creates an user
    user:
     name: '{{ item }}'
     create_home: True
     state: present
    loop:
     - myUserPlaybookA
     - myUserPlaybookB

  - name: My second task in the plabook creates a file in the user folder
    copy:
     content: "You must not share your password\n"
     dest: '{{ item }}'
    loop:
     - "/home/myUserPlaybookA/README.txt"
     - "/home/myUserPlaybookB/README.txt"

[ansibleuser@myCentosMachine ~]$
```

Variable especial para bucles: se sustituye por myUserPlaybookA y myUserPlaybookB

Tutorial Ansible

- Playbooks: iteraciones

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myCreateUser.yml
---

- name: This is my first ansible playbook
  hosts: myProductionEnvironment
  become: True
  tasks:
  - name: My first task
    user:
     name: '{{ item }}'
     create_home: True
     state: present
    loop:
     - myUserPlaybookA
     - myUserPlaybookB

  - name: My second task
    copy:
     content: "You must
     dest: '{{ item }}'
    loop:
     - "/home/myUserPlay
     - "/home/myUserPlay

[ansibleuser@myCentosMac
```

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible-playbook myCreateUser.yml

PLAY [This is my first ansible playbook] ***************************************************************

TASK [Gathering Facts] *********************************************************************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [My first task in the playbook creates an user] ***************************************************
changed: [myUbuntuMachine.local] => (item=myUserPlaybookA)
changed: [myCentosMachine.local] => (item=myUserPlaybookA)
changed: [myUbuntuMachine.local] => (item=myUserPlaybookB)
changed: [myCentosMachine.local] => (item=myUserPlaybookB)

TASK [My second task in the plabook creates a file in the user folder] *********************************
changed: [myCentosMachine.local] => (item=/home/myUserPlaybookA/README.txt)
changed: [myUbuntuMachine.local] => (item=/home/myUserPlaybookA/README.txt)
changed: [myCentosMachine.local] => (item=/home/myUserPlaybookB/README.txt)
changed: [myUbuntuMachine.local] => (item=/home/myUserPlaybookB/README.txt)

PLAY RECAP *********************************************************************************************
myCentosMachine.local      : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
myUbuntuMachine.local      : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansibleuser@myCentosMachine ~]$
```

Se ejecuta la tarea 2 veces por máquina (cambiando item por las que aparecen en el loop)

Tutorial Ansible

- Playbooks: iteraciones

ansibleuser@myCentosMachine:~

```
---

- name: This is my first ansible playbook
  hosts: myProductionEnvironment
  become: True
  tasks:
  - name: My first task in the playbook creates an user
    user:
     name: '{{ item }}'
     create_home: True
     state: present
    loop:
     - myUserPlaybookA
     - myUserPlaybookB

  - name: My second task in the plabook creates a file in the user folder
    copy:
     content: "You must not share your password\n"
     dest: '/home/{{ item }}/README.txt'
    loop:
     - myUserPlaybookA
     - myUserPlaybookB

[ansibleuser@myCentosMachine ~]$
```

Se concatena una variable con un String:
Primera iteración: /home/myUserPlaybookA/README.txt
Segunda iteración: /home/myUserPlaybookB/README.txt

Tutorial Ansible

- Playbooks: iteraciones

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myCreateUser.yml
---

- name: This is my first ansible playbook
  hosts: myProductionEnvironment
  become: True
  tasks:
  - name: My first task in the playbook creates an user
    user:
     name: '{{ item }}'
     create_home: True
     state: present
    loop: '{{ myUsers }}'
    vars:
     myUsers:
      - myUserPlaybookA
      - myUserPlaybookB

  - name: My second task in the plabook creates a file in the user folder
    copy:
     content: "You must not share your password\n"
     dest: '/home/{{ item }}/README.txt'
    loop: '{{ myUsers }}'
    vars:
     myUsers:
      - myUserPlaybookA
      - myUserPlaybookB

[ansibleuser@myCentosMachine ~]$
```

La tarea se ejecuta en bucle tomando los valores de la variable myUsers

Se puede crear una variable local a una tarea. La variable se llama myUsers y es una lista de dos elementos

Tutorial Ansible

- Playbooks: iteraciones

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myCreateUser.yml
---

- name: This is my first ansible playbook
  hosts: myProductionEnvironment
  become: True

  vars:
   myUsers:
    - myUserPlaybookA
    - myUserPlaybookB

  tasks:
  - name: My first task in the playbook creates an user
    user:
     name: '{{ item }}'
     create_home: True
     state: present
    loop: '{{ myUsers }}'

  - name: My second task in the plabook creates a file in the user folder
    copy:
     content: "You must not share your password\n"
     dest: '/home/{{ item }}/README.txt'
    loop: '{{ myUsers }}'
```

Se puede crear una variable global a todo el playbook. La variable se llama myUsers y es una lista de dos elementos

Tutorial Ansible

- Playbooks: iteraciones

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat ./vars/myVariables.yml
myUsers:
 - myUserPlaybookA
 - myUserPlaybookB
[ansibleuser@myCentosMachine ~]$
```

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myCreateUser.yml
---

- name: This is my first ansible playbook
  hosts: myProductionEnvironment
  become: True

  vars_files:
   - "./vars/myVariables.yml"

  tasks:
  - name: My first task in the playbook creates an user
    user:
     name: '{{ item }}'
     create_home: True
     state: present
    loop: '{{ myUsers }}'

  - name: My second task in the plabook creates a file in the user folder
    copy:
     content: "You must not share your password\n"
     dest: '/home/{{ item }}/README.txt'
    loop: '{{ myUsers }}'

[ansibleuser@myCentosMachine ~]$
```

Las variables se pueden guardar en archivos

Tutorial Ansible

- Playbooks: iteraciones

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat ./vars/myVariables.yml
myUsers:
 - myUserPlaybookA
 - myUserPlaybookB
[ansibleuser@myCentosMachine ~]$
```

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myCreateUser.yml
---

- name: This is my first ansible playbook
  hosts: myProductionEnvironment
  become: True

  tasks:
  - name: Load the variables of myVariables.yml
    include_vars:
     file: "./vars/myVariables.yml"

  - name: My first task in the playbook creates an user
    user:
     name: '{{ item }}'
     create_home: True
     state: present
    loop: '{{ myUsers }}'

  - name: My second task in the plabook creates a file in
    copy:
     content: "You must not share your password\n"
     dest: '/home/{{ item }}/README.txt'
    loop: '{{ myUsers }}'

[ansibleuser@myCentosMachine ~]$
```

Las variables se pueden guardar en archivos

Diferencias

- vars_file: lee las variables al comienzo del Playbook
- include_vars: lee las variables al comienzo de las tareas (puede que haya variables que no existían al comienzo del playbook).
  - También permite cargar variables de forma condicional ej. cárgame las variables de CentOS si la máquina es CentOS, etc

Tutorial Ansible

- Playbooks: iteraciones

Las variables se pueden sobrescribir en tiempo de ejecución -> tienen preferencia

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible-playbook myCreateUser.yml --extra-vars "{ myUsers: [myNewUserVar1, myNewUserV
ar2, myNewUserVar3] }"

PLAY [This is my first ansible playbook] ***************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [My first task in the playbook creates an user] ***************************************************************
changed: [myUbuntuMachine.local] => (item=myNewUserVar1)
changed: [myUbuntuMachine.local] => (item=myNewUserVar2)
changed: [myCentosMachine.local] => (item=myNewUserVar1)
changed: [myUbuntuMachine.local] => (item=myNewUserVar3)
changed: [myCentosMachine.local] => (item=myNewUserVar2)
changed: [myCentosMachine.local] => (item=myNewUserVar3)

TASK [My second task in the plabook creates a file in the user folder] *********************************************
changed: [myCentosMachine.local] => (item=myNewUserVar1)
changed: [myUbuntuMachine.local] => (item=myNewUserVar1)
changed: [myCentosMachine.local] => (item=myNewUserVar2)
changed: [myCentosMachine.local] => (item=myNewUserVar3)
changed: [myUbuntuMachine.local] => (item=myNewUserVar2)
changed: [myUbuntuMachine.local] => (item=myNewUserVar3)

PLAY RECAP *********************************************************************************************************
myCentosMachine.local      : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
myUbuntuMachine.local      : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansibleuser@myCentosMachine ~]$
```

Tutorial Ansible

- Playbooks: tipos de variables

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myVariablesTest.yml
---

- name: This is my first ansible playbook
  hosts: myProductionEnvironment
  become: True

  tasks:
   - name: Create number variable
     debug:
      msg: 'The numeric variable is {{ myNumVariable }}'
     vars:
      - myNumVariable: 8888

   - name: Create text variable
     debug:
      msg: 'The text variable is {{ myTextVariable }}'
     vars:
      - myTextVariable: myText

   - name: Create dict variable
     debug:
      msg: 'The name is {{ myDictVariable.myName }}, and the password is {{ myDictVariable.myPassword }}'
     vars:
      - myDictVariable:
         myName: dummyName
         myPassword: dummyPassW0rd


   - name: Create list variable
     debug:
      msg: 'The first value is {{ myListVariable[0] }}, and the second is {{ myListVariable[1] }}'
     vars:
      - myListVariable:
         - myFirstValue
         - mySecondValue

   - name: Use special variables
     debug:
      msg: 'The current host is {{ inventory_hostname }}'
```

Modulo para imprimir mensajes

Variable de tipo numérica

Variable de tipo texto

Variable de tipo diccionario

Variable de tipo lista

Variable especial (nos dice en qué máquina estamos)

Tutorial Ansible

- Playbooks: tipos d

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myVariablesTest.yml
---

- name: This is my first ansible playbook
  hosts: myProductionEnvironment
  become: True

  tasks:
   - name: Create number variable
     debug:
      msg: 'The numeric variable is {{ myNumVariable }}'
     vars:
      - myNumVariable: 8888

   - name: Create text variable
     debug:
      msg: 'The text variable is {{ myTextVariable }}'
     vars:
      - myTextVariable: myText

   - name: Create dict variable
     debug:
      msg: 'The name is {{ myDictVariable.myName }}, and th
     vars:
      - myDictVariable:
         myName: dummyName
         myPassword: dummyPassW0rd


   - name: Create list variable
     debug:
      msg: 'The first value is {{ myListVariable[0] }}, and
     vars:
      - myListVariable:
         - myFirstValue
         - mySecondValue

   - name: Use special variables
     debug:
      msg: 'The current host is {{ inventory_hostname }}'
```

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible-playbook myVariablesTest.yml

PLAY [This is my first ansible playbook] ***************************************

TASK [Gathering Facts] *********************************************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [Create number variable] **************************************************
ok: [myCentosMachine.local] => {
    "msg": "The numeric variable is 8888"
}
ok: [myUbuntuMachine.local] => {
    "msg": "The numeric variable is 8888"
}

TASK [Create text variable] ****************************************************
ok: [myCentosMachine.local] => {
    "msg": "The text variable is myText"
}
ok: [myUbuntuMachine.local] => {
    "msg": "The text variable is myText"
}

TASK [Create dict variable] ****************************************************
ok: [myCentosMachine.local] => {
    "msg": "The name is dummyName, and the password is dummyPassW0rd"
}
ok: [myUbuntuMachine.local] => {
    "msg": "The name is dummyName, and the password is dummyPassW0rd"
}

TASK [Create list variable] ****************************************************
ok: [myCentosMachine.local] => {
    "msg": "The first value is myFirstValue, and the second is mySecondValue"
}
ok: [myUbuntuMachine.local] => {
    "msg": "The first value is myFirstValue, and the second is mySecondValue"
}

TASK [Use special variables] ***************************************************
ok: [myCentosMachine.local] => {
    "msg": "The current host is myCentosMachine.local"
}
ok: [myUbuntuMachine.local] => {
    "msg": "The current host is myUbuntuMachine.local"
}

PLAY RECAP *********************************************************************
myCentosMachine.local      : ok=6    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
myUbuntuMachine.local      : ok=6    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansibleuser@myCentosMachine ~]$
```

Tutorial Ansible

- Playbooks: depuración
  - Muy útil cuando hay variables, bucles, etc

ansibleuser@myCentosMachine:~

```
[ansibleuser@myCentosMachine ~]$ cat myDebugPlayBook.yml
---

 - name: This is my first ansible playbook
   hosts: myProductionEnvironment

   tasks:
     - name: Load the variables of myVariables.yml
       include_vars:
        file: "./vars/myVariables.yml"

     - name: My first debug task
       debug:
        msg: "I want to create a user {{ item }} in the machine {{ inventory_hostname }}"
       loop: '{{ myUsers }}'

     - name: My second debug task
       debug:
         msg: "I want to create a file /home/{{ item }}/README.txt in the machine {{ inventory_hostname }}"
       loop: '{{ myUsers }}'

[ansibleuser@myCentosMachine ~]$
```

ansibleuser@myCentosMachine:~

```
[ansibleuser@myCentosMachine ~]$ ansible-playbook myDebugPlayBook.yml

PLAY [This is my first ansible playbook] ***************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [Load the variables of myVariables.yml] ***********************************************************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [My first debug task] *****************************************************************************************
ok: [myCentosMachine.local] => (item=myUserPlaybookA) => {
    "msg": "I want to create a user myUserPlaybookA in the machine myCentosMachine.local"
}
ok: [myCentosMachine.local] => (item=myUserPlaybookB) => {
    "msg": "I want to create a user myUserPlaybookB in the machine myCentosMachine.local"
}
ok: [myUbuntuMachine.local] => (item=myUserPlaybookA) => {
    "msg": "I want to create a user myUserPlaybookA in the machine myUbuntuMachine.local"
}
ok: [myUbuntuMachine.local] => (item=myUserPlaybookB) => {
    "msg": "I want to create a user myUserPlaybookB in the machine myUbuntuMachine.local"
}

TASK [My second debug task] ****************************************************************************************
ok: [myCentosMachine.local] => (item=myUserPlaybookA) => {
    "msg": "I want to create a file /home/myUserPlaybookA/README.txt in the machine myCentosMachine.local"
}
ok: [myCentosMachine.local] => (item=myUserPlaybookB) => {
    "msg": "I want to create a file /home/myUserPlaybookB/README.txt in the machine myCentosMachine.local"
}
ok: [myUbuntuMachine.local] => (item=myUserPlaybookA) => {
    "msg": "I want to create a file /home/myUserPlaybookA/README.txt in the machine myUbuntuMachine.local"
}
ok: [myUbuntuMachine.local] => (item=myUserPlaybookB) => {
    "msg": "I want to create a file /home/myUserPlaybookB/README.txt in the machine myUbuntuMachine.local"
}

PLAY RECAP *********************************************************************************************************
myCentosMachine.local      : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
myUbuntuMachine.local      : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansibleuser@myCentosMachine ~]$
```

Tutorial Ansible

- Playbooks: registrar la salida de una tarea
  - Guarda el resultado de una salida en una variable

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myRegisterPlaybook.yml
---

 - name: This playbook shows how to register the result of a task
   hosts: myProductionEnvironment

   tasks:
     - name: Print some information
       shell:
         cmd: echo hello worl!
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "The result of the previous command is: {{ myCommandOutput }}"

[ansibleuser@myCentosMachine ~]$
```

Se ejecuta el comando “echo hello world!“ en cada máquina de myProductionEnvironment

Se guarda el resultado de la ejecución de la tarea.
En el caso del módulo Shell:

Return Values

Common return values are documented here, the following are the fields unique to this module:

Key	Returned	Description
cmd string	always	The command executed by the task. **Sample:** rabbitmqctl join_cluster rabbit@master
delta string	always	The command execution delta time. **Sample:** 0:00:00.325771
end string	always	The command execution end time. **Sample:** 2016-02-25 09:18:26.755339
msg	always	changed

Tutorial Ansible

- Playbooks: registrar la salida de una tarea
  - Guarda el resultado de una salida en una variable

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myRegisterPlaybook.yml
---

 - name: This playbook shows how to register the result of a task
   hosts: myProductionEnvironment

   tasks:
     - name: Print some information
       shell:
         cmd: echo hello worl!
       register: myCommandOutput

     - name: Obtain the output of the
       debug:
        msg: "The result of the previ

[ansibleuser@myCentosMachine ~]$
```

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible-playbook myRegisterPlaybook.yml

PLAY [This playbook shows how to register the result of a task] ************************************************

TASK [Gathering Facts] *****************************************************************************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [Print some information] **********************************************************************************
changed: [myCentosMachine.local]
changed: [myUbuntuMachine.local]

TASK [Obtain the output of the previous execution] *************************************************************
ok: [myCentosMachine.local] => {
    "msg": "The result of the previous command is: {'cmd': 'echo hello worl!', 'stdout': 'hello worl!', 'stderr': '',
'rc': 0, 'start': '2021-02-14 17:38:06.071776', 'end': '2021-02-14 17:38:06.080482', 'delta': '0:00:00.008706', 'chang
ed': True, 'stdout_lines': ['hello worl!'], 'stderr_lines': [], 'failed': False}"
}
ok: [myUbuntuMachine.local] => {
    "msg": "The result of the previous command is: {'cmd': 'echo hello worl!', 'stdout': 'hello worl!', 'stderr': '',
'rc': 0, 'start': '2021-02-14 17:38:06.262990', 'end': '2021-02-14 17:38:06.265831', 'delta': '0:00:00.002841', 'chang
ed': True, 'stdout_lines': ['hello worl!'], 'stderr_lines': [], 'failed': False}"
}

PLAY RECAP *****************************************************************************************************
myCentosMachine.local      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
myUbuntuMachine.local      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansibleuser@myCentosMachine ~]$
```

Tutorial Ansible

- Playbooks: registrar la salida de una tarea
  - Guarda el resultado de una salida en una variable

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myRegisterPlaybook.yml
---

 - name: This playbook shows how to register the result of a task
   hosts: myProductionEnvironment

   tasks:
     - name: Print some information
       shell:
         cmd: echo hello worl!
       register: myCommandOutput

     - name: Obtain only the standard output of the execution
       debug:
        msg: "The output of the command is: {{ myCommandOutput.stdout_lines }}"

[ansibleuser@myCentosMachine ~]$
```

Se puede acceder a un campo del resultado registrado

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible-playbook myRegisterPla

PLAY [This playbook shows how to register the result of a task]

TASK [Gathering Facts] *********************************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [Print some information] **************************************
changed: [myCentosMachine.local]
changed: [myUbuntuMachine.local]

TASK [Obtain only the standard output of the execution] ********
ok: [myCentosMachine.local] => {
    "msg": "The output of the command is: ['hello worl!']"
}
ok: [myUbuntuMachine.local] => {
    "msg": "The output of the command is: ['hello worl!']"
}

PLAY RECAP *********************************************************
myCentosMachine.local      : ok=3    changed=1    unreachable=0
myUbuntuMachine.local      : ok=3    changed=1    unreachable=0

[ansibleuser@myCentosMachine ~]$
```

Tutorial Ansible

- Playbooks: variables de entorno

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myEnvironmentVariable.yml
---

 - name: This playbook shows how to create and use environmental variables
   hosts: myProductionEnvironment

   tasks:
     - name: Check that there is no environmental variable
       shell:
         cmd: echo my java home is in $JAVA_HOME
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "Output: {{ myCommandOutput.stdout_lines }}"


     - name: Create environmental variable
       shell:
         cmd: echo my java home is in $JAVA_HOME
       environment:
         JAVA_HOME: /usr/local/java-1.8
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "Output: {{ myCommandOutput.stdout_lines }}"


     - name: Check that the environmental variable is not still live
       shell:
         cmd: echo my java home is in $JAVA_HOME
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "Output: {{ myCommandOutput.stdout_lines }}"
```

Se crea una variable de entorno en cada una de las máquinas de myProductionEnvironment **sólo para esa tarea**

Tutorial Ansible

- Playbooks: variables de entorno

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myEnvironmentVariable.yml
---

 - name: This playbook shows how to create and use environmental variables
   hosts: myProductionEnvironment

   tasks:
     - name: Check that there is no environmental variable
       shell:
         cmd: echo my java home is in $JAVA_HOME
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "Output: {{ myCommandOutput.stdout_lines }}"


     - name: Create environmental variable
       shell:
         cmd: echo my java home is in $JAVA_HOME
       environment:
         JAVA_HOME: /usr/local/java-1.8
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "Output: {{ myCommandOutput.stdout_lines }}"


     - name: Check that the environmental variable is not still live
       shell:
         cmd: echo my java home is in $JAVA_HOME
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "Output: {{ myCommandOutput.stdout_lines }}"
```

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible-playbook myEnvironmentVariable.yml

PLAY [This playbook shows how to create and use environmental variables] *****

TASK [Gathering Facts] *********************************************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [Check that there is no environmental variable] ***************************
changed: [myCentosMachine.local]
changed: [myUbuntuMachine.local]

TASK [Obtain the output of the previous execution] *****************************
ok: [myCentosMachine.local] => {
    "msg": "Output: ['my java home is in']"
}
ok: [myUbuntuMachine.local] => {
    "msg": "Output: ['my java home is in']"
}

TASK [Create environmental variable] *******************************************
changed: [myCentosMachine.local]
changed: [myUbuntuMachine.local]

TASK [Obtain the output of the previous execution] *****************************
ok: [myCentosMachine.local] => {
    "msg": "Output: ['my java home is in /usr/local/java-1.8']"
}
ok: [myUbuntuMachine.local] => {
    "msg": "Output: ['my java home is in /usr/local/java-1.8']"
}

TASK [Check that the environmental variable is not still live] *****************
changed: [myCentosMachine.local]
changed: [myUbuntuMachine.local]

TASK [Obtain the output of the previous execution] *****************************
ok: [myCentosMachine.local] => {
    "msg": "Output: ['my java home is in']"
}
ok: [myUbuntuMachine.local] => {
    "msg": "Output: ['my java home is in']"
}

PLAY RECAP *********************************************************************
myCentosMachine.local      : ok=7    changed=3    unreachable=0    failed=0
myUbuntuMachine.local      : ok=7    changed=3    unreachable=0    failed=0

[ansibleuser@myCentosMachine ~]$
```

structura como
J Morán

Tutorial Ansible

- Playbooks: variables de entorno

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myEnvironmentVariableInPlaybook.yml
---

 - name: This playbook shows how to create and use environmental variables
   hosts: myProductionEnvironment

   environment:
     JAVA_HOME: /usr/local/java-1.8

   tasks:
     - name: Check that there is an environmental variable
       shell:
         cmd: echo my java home is in $JAVA_HOME
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "Output: {{ myCommandOutput.stdout_lines }}"


     - name: Check that the environmental variable is still live
       shell:
         cmd: echo my java home is in $JAVA_HOME
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "Output: {{ myCommandOutput.stdout_lines }}"

[ansibleuser@myCentosMachine ~]$
```

Se crea una variable de entorno en cada una de las máquinas de myProductionEnvironment **para todo el playbook**

o código

Tutorial Ansible

- Playbooks: variables de entorno

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myEnvironmentVariableInPlaybook.yml
---

 - name: This playbook shows how to create and use environmental variables
   hosts: myProductionEnvironment

   environment:
     JAVA_HOME: /usr/local/java-1.8

   tasks:
     - name: Check that there is an environmental variable
       shell:
         cmd: echo my java home is in $JAVA_HOME
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "Output: {{ myCommandOutput.stdout_lines }}"


     - name: Check that the environmental variable is still live
       shell:
         cmd: echo my java home is in $JAVA_HOME
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "Output: {{ myCommandOutput.stdout_lines }}"


[ansibleuser@myCentosMachine ~]$
```

La variable de entorno no sigue viva después de finalizar el playbook

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ ansible-playbook myEnvironmentVariableInPlaybook.yml

PLAY [This playbook shows how to create and use environmental variables] **************

TASK [Gathering Facts] ********************************************************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [Check that there is an environmental variable] **************************************
changed: [myCentosMachine.local]
changed: [myUbuntuMachine.local]

TASK [Obtain the output of the previous execution] ****************************************
ok: [myCentosMachine.local] => {
    "msg": "Output: ['my java home is in /usr/local/java-1.8']"
}
ok: [myUbuntuMachine.local] => {
    "msg": "Output: ['my java home is in /usr/local/java-1.8']"
}

TASK [Check that the environmental variable is still live] ********************************
changed: [myCentosMachine.local]
changed: [myUbuntuMachine.local]

TASK [Obtain the output of the previous execution] ****************************************
ok: [myCentosMachine.local] => {
    "msg": "Output: ['my java home is in /usr/local/java-1.8']"
}
ok: [myUbuntuMachine.local] => {
    "msg": "Output: ['my java home is in /usr/local/java-1.8']"
}

PLAY RECAP ********************************************************************************
myCentosMachine.local      : ok=5    changed=2    unreachable=0    failed=0    skipped=0
myUbuntuMachine.local      : ok=5    changed=2    unreachable=0    failed=0    skipped=0
```

Tutorial Ansible

- Playbooks: variables de entorno

```
ansibleuser@myCentosMachine:~
[ansibleuser@myCentosMachine ~]$ cat myGlobalEnvironmentalVariable.yml
---

 - name: This playbook shows how to create a global environmental variable
   hosts: myProductionEnvironment

   tasks:
     - name: Check that there is not  an environmental variable
       shell:
         cmd: echo my java home is in $JAVA_HOME
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "Output: {{ myCommandOutput.stdout_lines }}"



     - name: Create the java home variable
       lineinfile:
         dest: ~/.bashrc
         regexp: '^JAVA_HOME'
         line: 'JAVA_HOME=/usr/local/java-1.8'
         state: present

     - name: Use the environmental variable with sourcing the environmental variables
       shell:
         cmd: . ~/.bashrc && echo my java home is in $JAVA_HOME
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "Output: {{ myCommandOutput.stdout_lines }}"



     - name: Check that the environmental variable can not be used in playbook without sourcing
       shell:
         cmd: echo my java home is in $JAVA_HOME
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "Output: {{ myCommandOutput.stdout_lines }}"
```

Crea una variable de entorno de forma permanente en bashrc
lineinfile: permite tener o no tener líneas en archivos
Si alguna línea comienza por JAVA_HOME, la cambia por JAVA_HOME=/usr/local/java-1.8 (y sino la añade)

Para poder utilizarla en el playbook se tiene que hace un source

omo código
ın

Tutorial Ansible

La variable de entorno sigue viva después de finalizar el playbook

- Playbooks: varia

ansibleuser@myCentosMachine:~

```
[ansibleuser@myCentosMachine ~]$ cat myGlobalEnvironmentalVariab
---

 - name: This playbook shows how to create a global environmenta
   hosts: myProductionEnvironment

   tasks:
     - name: Check that there is not  an environmental variable
       shell:
         cmd: echo my java home is in $JAVA_HOME
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "Output: {{ myCommandOutput.stdout_lines }}"



     - name: Create the java home variable
       lineinfile:
         dest: ~/.bashrc
         regexp: '^JAVA_HOME'
         line: 'JAVA_HOME=/usr/local/java-1.8'
         state: present

     - name: Use the environmental variable with sourcing the en
       shell:
         cmd: . ~/.bashrc && echo my java home is in $JAVA_HOME
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "Output: {{ myCommandOutput.stdout_lines }}"



     - name: Check that the environmental variable can not be us
       shell:
         cmd: echo my java home is in $JAVA_HOME
       register: myCommandOutput

     - name: Obtain the output of the previous execution
       debug:
        msg: "Output: {{ myCommandOutput.stdout_lines }}"
```

ansibleuser@myCentosMachine:~

```
[ansibleuser@myCentosMachine ~]$ ansible-playbook myGlobalEnvironmentalVariable.yml

PLAY [This playbook shows how to create a global environmental variable] ********************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [myCentosMachine.local]
ok: [myUbuntuMachine.local]

TASK [Check that there is not  an environmental variable] **********************************************************
changed: [myCentosMachine.local]
changed: [myUbuntuMachine.local]

TASK [Obtain the output of the previous execution] *****************************************************************
ok: [myCentosMachine.local] => {
    "msg": "Output: ['my java home is in']"
}
ok: [myUbuntuMachine.local] => {
    "msg": "Output: ['my java home is in']"
}

TASK [Create the java home variable] *******************************************************************************
changed: [myCentosMachine.local]
changed: [myUbuntuMachine.local]

TASK [Use the environmental variable with sourcing the environmental variables] ************************************
changed: [myCentosMachine.local]
changed: [myUbuntuMachine.local]

TASK [Obtain the output of the previous execution] *****************************************************************
ok: [myCentosMachine.local] => {
    "msg": "Output: ['my java home is in /usr/local/java-1.8']"
}
ok: [myUbuntuMachine.local] => {
    "msg": "Output: ['my java home is in']"
}

TASK [Check that the environmental variable can not be used in playbook without sourcing] **************************
changed: [myCentosMachine.local]
changed: [myUbuntuMachine.local]

TASK [Obtain the output of the previous execution] *****************************************************************
ok: [myCentosMachine.local] => {
    "msg": "Output: ['my java home is in']"
}
ok: [myUbuntuMachine.local] => {
    "msg": "Output: ['my java home is in']"
}

PLAY RECAP *********************************************************************************************************
myCentosMachine.local      : ok=8    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
myUbuntuMachine.local      : ok=8    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansibleuser@myCentosMachine ~]$
```

Tutorial Ansible

- Playbooks: otro
  - Bucles: se puede acceder al índice, pausarlos, anidarlos, iterar sobre el inventario
  - Until: permite crear tareas repetidas
  - Condicionales: si es Ubuntu instala apache2, sino httpd
  - Aleatorio: random
  - Tags: permiten poner etiquetas a las tareas para luego indicar con las tags qué tareas queremos ejecutar
  - Handlers: las tareas pueden notificar que se ejecute un handler. Ejemplo: si se modifica alguna propiedad de httpd, entonces reinicia el servicio (el módulo notifica que se tiene que ejecutar el handle de reinicio)
  - Roles: agrupaciones de tareas, variables, etc que indican el estado al que tiene que llegar una máquina. Ej. rol de PostgreSQL

Jesús Morán

Grupo de Investigación en Ingeniería del Software

http://giis.uniovi.es

Universidad de Oviedo