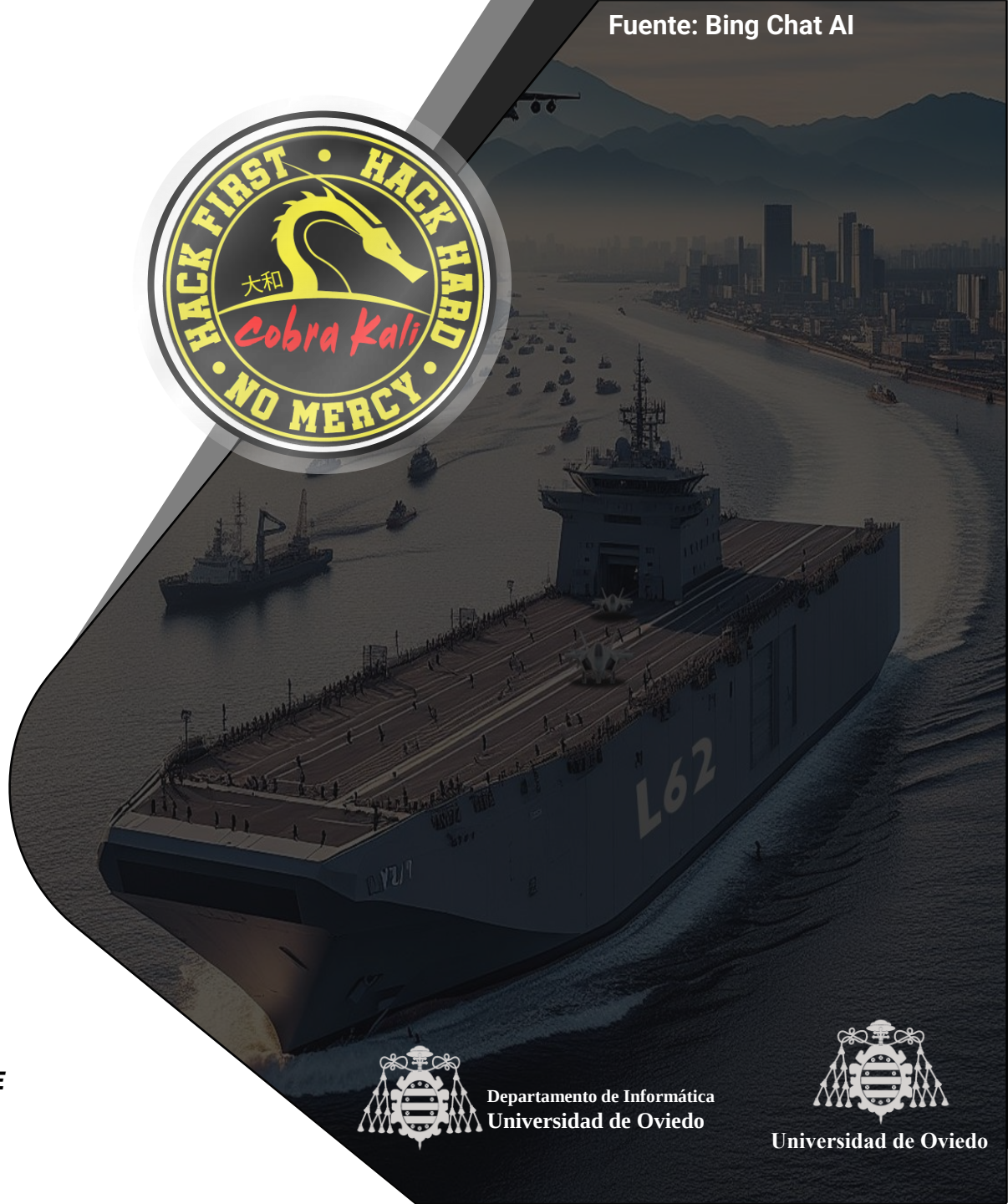
INTRODUCCIÓN A ANSIBLE

Controlando sistemas de forma remota

José Manuel Redondo López Proyecto "*L-62 Princesa de Asturias*" v1.2

Fuente: Bing Chat AI

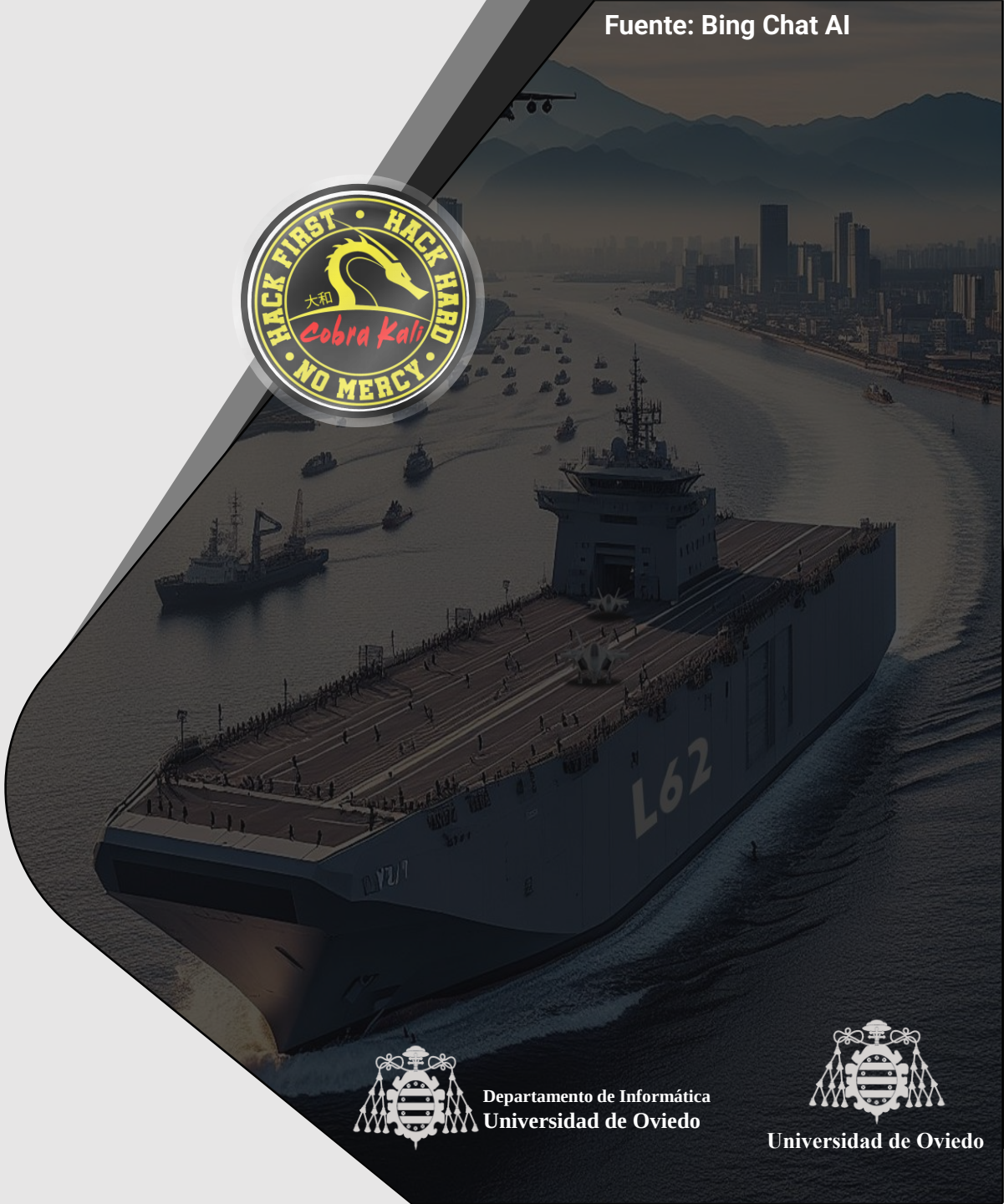


Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

ÍNDICE

< Ir al Índice



Fuente: Bing Chat AI

INTRODUCCIÓN

¿Qué es lo que vamos a ver?

Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

¿Qué es Ansible?

José Manuel
Redondo López

- **La tradicional separación entre administradores y desarrolladores está tendiendo a desaparecer**
  - La creación de infraestructuras, su administración y la instalación de los programas que hacen falta en las mismas está transformándose
  - **Ahora todo se hace vía código**
- **Esto permite enfrentarse a escenarios mucho más complejos**
  - Integrar estas operaciones **en el pipeline de desarrollo** de un producto
  - Ser capaz de reproducir una infraestructura exactamente cuando sea necesario,
  - ¡Incluyendo todo su software!
- **Esto es lo que tradicionalmente se conoce como DevOps**
  - Integrar en una misma unidad las **labores de desarrollo con las de creación de infraestructura** y su administración
  - Vagrant, LXD y Docker son formas de crear infraestructura
  - Ansible es una forma de **administrarla**

¿Qué es Ansible?

José Manuel
Redondo López

- **Por tanto, es una herramienta de administración que permite la configuración y aprovisionamiento automático**
  - Permite establecer la configuración de un sistema (qué se instala y cómo) **con código de un programa** en un lenguaje especial: Domain-Specific Language o DSL
  - Reemplaza la administración con scripts y **facilita migrar los existentes** a Ansible poco a poco
- **Es uno de los más fáciles de empezar a usar**
  - Desarrollado por Red Hat ®
  - Y actualmente es una herramienta **muy popular** en la empresa privada para ello
  - Sirve además de buena **base para entender otras similares**
- **Solo necesita SSH para conectarse a los servidores y ejecutar las tareas configuradas**
  - ¡No necesita servidores ni daemons!
  - Es muy fácil convertir scripts bash en tareas de Ansible
  - Ansible termina ejecutando los mismos comandos que un bash script
  - Esta es otra razón por la que es tan popular

¿Qué es Ansible?

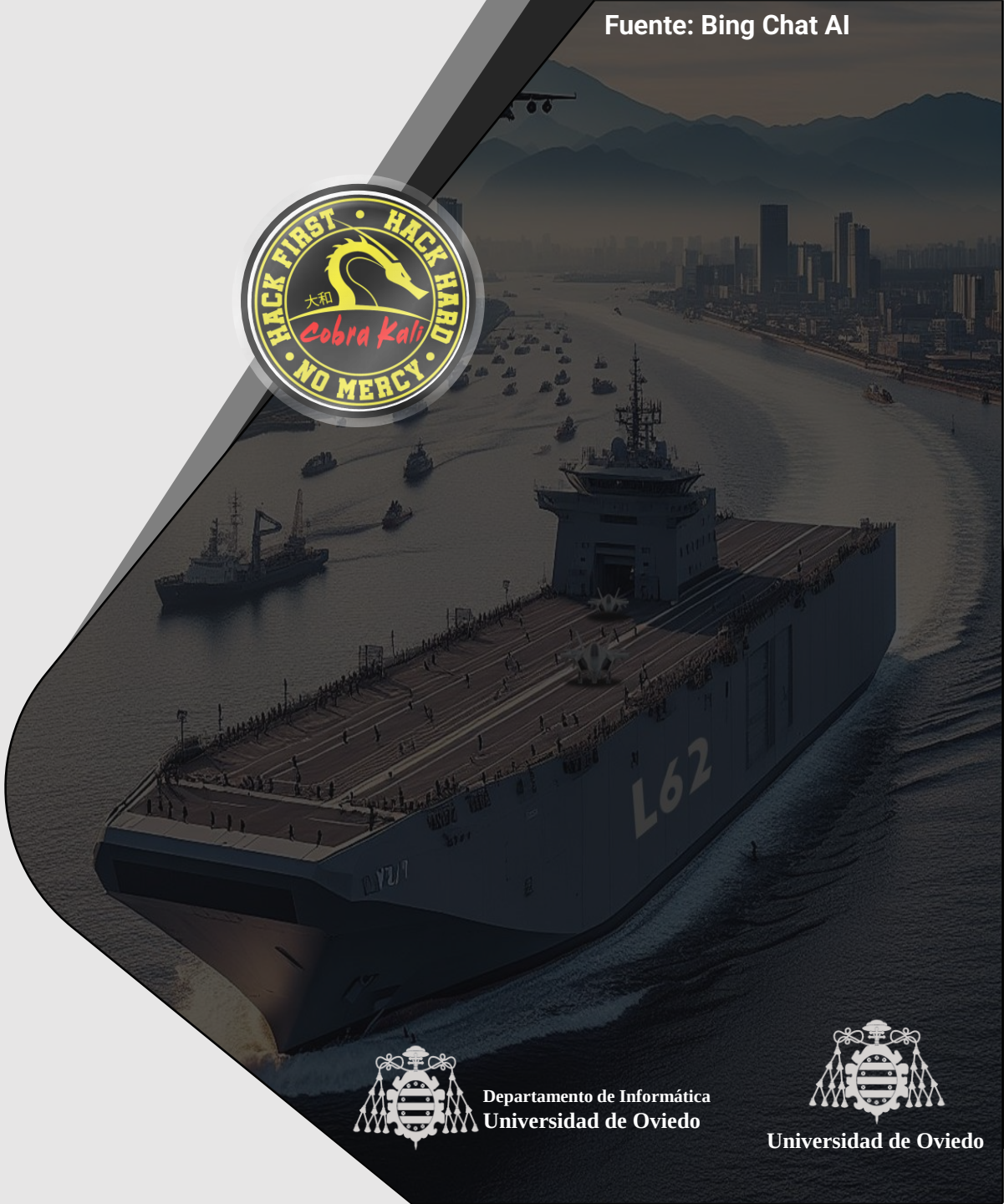
José Manuel
Redondo López

- **Automatiza el proceso de obtener el contexto del servidor antes de ejecutar tareas**
  - **Qué recursos tiene**, sus ficheros, los procesos que ejecuta…
    - Toda clase de información necesaria para hacer las tareas de administración
  - A cada trozo de información de contexto de la denomina **Fact**
  - Debido a ello, es capaz de manejar la mayoría de los casos complejos con scripts simples
- **Las tareas de Ansible son idempotentes**
  - Su resultado es siempre el mismo, ¡no importa cuantas veces se ejecuten!
  - Se puede **ejecutar con seguridad una misma tarea varias veces** sin miedo a que rompa cosas ya hechas
    - Esto por ejemplo no es sencillo de conseguir con un script bash…
    - Utiliza las facts para comprobar el estado de cada recurso y ver si necesita cambiar algo para obtener el resultado deseado
  - ¿El sistema ya está como indica el script Ansible? Entonces no hago **NADA**

< Ir al Índice

INSTALACIÓN

Como incorporar Ansible al software de una máquina



Fuente: Bing Chat AI

Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

¿QUÉ VAMOS A VER EN ESTE APARTADO?

José Manuel
Redondo López

- **Una serie de conceptos que cubren la instalación y puesta en marcha de Ansible**
  - **Cómo se instala** y las bases de su funcionamiento
  - Qué es un **inventario**, como se crea y para qué sirve
  - Cómo dar las **primeras órdenes de prueba** a las máquinas manejadas

Instalación de Ansible global

José Manuel
Redondo López

- Antes de ejecutar tareas Ansible es necesario instalar el software necesario
- **Ansible no tiene "agentes" a instalar en las máquinas a controlar**
  - Las máquinas remotas se pueden controlar **mediante un simple acceso vía SSH**
  - Por tanto, se puede ejecutar en cualquier servidor Linux (no Windows)
  - Y puede controlar otro tipo de clientes (Linux, Windows) **sin necesidad de modificarlos**
- **La instalación en Ubuntu vía paquetes lo instala globalmente (todos los usuarios)**
  - La configuración por defecto de un Ansible instalado globalmente está en `/etc/ansible`
  - Será la opción que hagamos en este curso, por simplicidad
- **Pero podemos instalar Ansible sólo para usuarios concretos**
  - Usando `pip` para instalar `virtualenv` de Python
  - Permite que puedan usarlo **solo usuarios autorizados**
  - Además de **probar nuevas versiones aisladamente** sin afectar a la instalada globalmente
  - https://www.redhat.com/sysadmin/python-venv-ansible

Inventory

José Manuel
Redondo López

- **Al instalar Ansible se crea un inventory file donde se definen los servidores que va a manejar**
  - En `/etc/ansible/hosts`
  - Se puede llamar de cualquier forma, pero lo dejaremos con su nombre por defecto
- **Lista las direcciones de las máquinas manejadas y su etiqueta**
  - La etiqueta puede ser la que queramos
  - Pero es mejor **usar alguna que describa sus funciones**
    - La propia máquina puede aparecer bajo `[local]`
    - Otros servidores web por `[nginx_web]`
  - Permite **agrupar máquinas** a controlar por función
  - Debemos asegurarnos de que **hay conectividad** con ellas
  - Se pueden hacer **grupos de grupos** de máquinas ☺

```
GNU nano 2.9.3                    hosts                    M

# Ex 3: A collection of database servers in the 'dbservers' group

## [dbservers]
##
## db01.intranet.mydomain.net
## db02.intranet.mydomain.net
## 10.25.1.56
## 10.25.1.57

# Here's another example of host ranges, this time there are no
# leading 0s:

## db-[99:101]-node.example.com

[local]
127.0.0.1

[nginx_web]
192.168.1.200
192.168.1.201
```

Inventario sencillo de Ansible

INVENTORY

José Manuel
Redondo López

- El archivo `hosts` por defecto viene con varios ejemplos de cómo hacer esto
- Además de este inventory simple, podemos hacer otras cosas
  - Establecer **rangos de hosts** que nos permitan manipular varios automáticamente
  - Podemos **usar un inventario concreto** en lugar del general (es el que se usa por defecto)
    - Basta con añadir la opción `-i <ruta al fichero con el inventory>` **a cualquier comando** que veremos en esta presentación
  - La ejecución de Ansible entre los diferentes servidores de un inventario se hace en paralelo
    - Es decir, **no es determinista** (no puedes saber el orden exacto)
- Pero existen opciones mucho más avanzadas
  - Pueden verse otras formas de configurar el inventory aquí
    - https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html
  - Y también puede consultarse como hacer un inventory dinámico aquí
    - https://docs.ansible.com/ansible/latest/user_guide/intro_dynamic_inventory.html

IMPORTANCIA DEL INVENTORY

José Manuel
Redondo López

- **El inventario es lo que define qué endpoints (máquinas) va a manejar Ansible**
  - Es muy importante que lo definas correctamente
- **Hay dos formatos para definir inventarios**
  - El **INI** (el anterior) y el **YAML**
  - Ambos son equivalentes
  - El YAML quizá sea más fácil de entender y usar cuando hay muchos endpoints
- **Los inventarios también pueden ser dinámicos (definidos por código)**

Inventario de Ansible en dos formatos distintos

Formato INI

```
uniovi.es
[facultades]
eii.uniovi.es
epi.uniovi.es
[BBDDS]
cassi.uniovi.es
onedrive.uniovi.es
```

Formato YAML

```
ALL:
  HOSTS:
    UNIOVI.ES:
  CHILDREN:
    FACULTADES:
      HOSTS:
        EII.UNIOVI.ES:
        EPI.UNIOVI.ES:
    BBDDS:
      HOSTS:
        CASSI.UNIOVI.ES:
        ONEDRIVE.UNIOVI.ES:
```

Sobre la definición de inventorys

José Manuel
Redondo López

- **Hay dos grupos de hosts por defecto: `all` y `ungrouped`**
  - `all` contiene todos los hosts
  - `ungrouped` contiene todos los hosts **que no pertenecen** a un grupo que no sea `all`
- **Se puede poner un mismo host en más de un grupo**
- **Dentro de un grupo puedes especificar como `children` otros grupos ya definidos**
- **Si tienes nombres de hosts que siguen un mismo patrón, pueden añadirse como un rango en lugar de listarlos individualmente**
  - Por ejemplo, en formato YAML:

```
webservers:
   hosts:
     www[01:10].uniovi.es:
```

  - Y el equivalente en formato INI:

```
[webservers]
www[01:10].uniovi.es
```

Primeros pasos

José Manuel
Redondo López

- El verdadero poder de Ansible se demuestra realmente manejando servidores remotos, no el propio local
- Pero para ello debemos asegurarnos de lo siguiente
  - Tener servidores en las IPs indicadas anteriormente
  - Que dichos servidores **admitan conexiones SSH** desde el servidor que tiene Ansible instalado
  - **Preferiblemente de forma automática**
    - Es decir, **con certificados**, sin passwords
- Una conexión local (manejar con Ansible tu propia máquina) no necesita acceso a SSH

Conceptos básicos: Ejecutar comandos

José Manuel
Redondo López

- Durante las conexiones, es posible que aparezca el error “Too many authenticacion failures”
- En este caso, la razón será seguramente que la conexión automática vía SSH no está configurada u operativa
  - Me refiero a la **conexión SSH usando certificados**, la más aconsejable en estos escenarios
- En ese caso, podemos hacer que se nos pida una contraseña para iniciar sesión con
  - `ansible --ask-pass --ssh-extra-args-'-o "PubkeyAuthentication-no“’ nginx_web -m ping`

¿Crees que lo has entendido todo? ¡Autoevalúate!

José Manuel
Redondo López

- *¿Has entendido que una de las características más interesantes de **Ansible** es que no requiere instalar nada en las máquinas a controlar?*
- *¿Comprendes que solo necesitas un acceso vía SSH a las máquinas a controlar, con un usuario que tenga permisos suficientes para hacer las acciones que les pidas?*
- *¿Entiendes que no tienes que instalar **Ansible** para todos los usuarios, sino que se lo puedes instalar a uno solo?*
- *¿Entiendes el propósito de un inventario, para que sirve y que realmente puedes crear inventarios independientes cuando quieras?*
- *¿Eres consciente de que la verdadera utilidad de **Ansible** es hacer grupos de máquinas a controlar en bloque?*
- *¿Comprendes que los inventarios se pueden definir usando dos sintaxis distintas?*
- *¿Entiendes que hay grupos especiales y que puedes anidar grupos con grupos hijo?*
- *¿Has entendido como mandar una orden de ejemplo con **Ansible** a una máquina o grupo de máquinas remoto?*

< Ir al Índice

Fuente: Bing Chat AI

Facts >

Módulos >

Playbooks, Tasks ... >

Roles y Templates >

Conceptos Principales de Ansible

Trabajando con los elementos que componen el producto

Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

¿QUÉ VAMOS A VER EN ESTE APARTADO?

José Manuel
Redondo López

- **En este apartado vamos a hablar de los primeros conceptos principales de Ansible: Facts y módulos**
  - La **información que se puede sacar** de las máquinas remotas gracias a Ansible
  - Cómo lo que Ansible puede hacer se puede **expandir enormemente gracias a la biblioteca de módulos** que tiene
  - El **tratamiento general** que debe hacerse a los módulos y su documentación
  - Como **ejecutar algunos módulos** básicos interesantes

Facts

Datos de una máquina

Facts

José Manuel
Redondo López

- **Antes de ejecutar cualquier tarea, Ansible recopilará información sobre el sistema que está aprovisionando**
  - Esa información se usa para ejecutar la tarea con más garantías de que termine correctamente
- **Cada pieza de información se llama Fact**
- **Proporcionan una amplia gama de información del sistema**
  - Número de núcleos de CPU
  - Redes IPv4 e IPv6 disponibles
  - Discos montados
  - Tipo de SSOO
  - …
- **Un listado de esta información puede verse aquí**
  - https://docs.ansible.com/ansible/latest/user_guide/playbooks_vars_facts.html

Facts

José Manuel
Redondo López

- Las facts las podremos usar de dos formas
  - **Automáticamente**: Ansible las usa para averiguar si una tarea ya ha sido hecha
  - **Como variables en nuestros programas**: Cada fact es una variable cuyo valor podemos usar para tomar decisiones
- Un listado de todas se puede obtener con
  - `ansible <host> -m ansible.builtin.setup`


```
    },
    "ansible_processor": [
        "0",
        "AuthenticAMD",
        "AMD Ryzen 7 5800H with Radeon Graphics",
        "1",
        "AuthenticAMD",
        "AMD Ryzen 7 5800H with Radeon Graphics",
        "2",
        "AuthenticAMD",
        "AMD Ryzen 7 5800H with Radeon Graphics",
        "3",
        "AuthenticAMD",
        "AMD Ryzen 7 5800H with Radeon Graphics"
    ],
    "ansible_processor_cores": 4,
    "ansible_processor_count": 1,
    "ansible_processor_threads_per_core": 1,
    "ansible_processor_vcpus": 4,
    "ansible_product_name": "VirtualBox",
    "ansible_product_serial": "NA",
    "ansible_product_uuid": "NA",
    "ansible_product_version": "1.2",
    "ansible_python": {
        "executable": "/usr/bin/python3",
        "has_sslcontext": true,
        "type": "cpython",
        "version": {
            "major": 3,
            "micro": 10,
            "minor": 8,
            "releaselevel": "final",
            "serial": 0
        },
        "version_info": [
            3,
            8,
            10,
            "final",
            0
        ]
    },
    "ansible_python_version": "3.8.10",
```


Algunas facts de una máquina remota

Módulos

Organización de lo que hacemos con Ansible

Módulos

José Manuel
Redondo López

- **Ansible** usa **módulos** para hacer la mayoría de sus tareas
  - Pueden hacer cosas como instalar software, copiar archivos, usar plantillas…
  - Existe una **enorme cantidad de módulos** para hacer muchísimas cosas
  - https://docs.ansible.com/ansible/latest/modules/modules_by_category.html
- **Son la forma adecuada de utilizar Ansible**
  - Acceden al contexto de la máquina ("Facts") para determinar las acciones que faltan para realizar una tarea
    - **¡Sin repetir lo que ya está hecho!**
  - **Es decir, no ejecuta tareas innecesarias**
  - El código está revisado y validado: nos puede ahorrar **MUCHO** tiempo
  - Tienen parámetros para que se adapten a casos particulares
  - Si los usamos, lo que estaríamos haciendo es ejecutar comandos `bash` estándar

Module Index

- All modules
- Cloud modules
- Clustering modules
- Commands modules
- Crypto modules
- Database modules
- Files modules
- Identity modules
- Inventory modules
- Messaging modules
- Monitoring modules
- Net Tools modules
- Network modules
- Notification modules
- Packaging modules
- Remote Management modules
- Source Control modules
- Storage modules
- System modules
- Utilities modules
- Web Infrastructure modules
- Windows modules

Fuente: https://docs.ansible.com/ansible/2.9/modules/modules_by_category.html

Versatilidad de Ansible y sus Módulos

José Manuel
Redondo López

- **Cloud**
  - Tareas para AWS, CloudFormation, EC2, Lambda, S3, Azure, Docker, Google Cloud, Heroku, Linode, OpenStack, VMWare, Rackspace…
- **Clustering**
  - Tareas para Kubernetes, Consul, ETCD3, Pacemaker y Znode
- **Commands**
  - Ejecución de comandos en máquinas remotas, tanto Linux como Windows
- **Crypto**
  - Obtención de certificados para servidores, tanto con OpenSSL como con el protocolo ACME (Let's Encrypt)
- **Database**
  - Algunas tareas para interactuar con MongoDB, SQL Server, MySQL, PostgreSQL, …
- **Files**
  - Manejo general de archivos: acceso a listas de permisos, contenido, copia, encontrar texto, lectura (incluido CSVs), modificación, rsync…
- **Identity**
  - Manejo de claves con 1Password, CyberARK, FreeIPA, Keycloack y OpenDJ
- **Inventory**
  - Manejo con Ansible del propio inventario de Ansible ☺

VERSATILIIDAD DE ANSIBLE Y SUS MÓDULOS

José Manuel
Redondo López

- **Messaging**
  - Manejo del sistema de mensajería RabbitMQ (mencionado en el tema de colas anteriormente)
- **Monitoring**
  - Manejo de sistemas de monitorización DataDog, Grafana, LogicMonitor, Nagios, Pingdom, Sensu, Zabbix…
- **Net Tools**
  - Son diversas herramientas para servicios de red como DNS, Haproxy, Geolocalización, Cloudflare, SNMP, descarga de URLs, LDAP, NIOS…
- **Network**
  - Manejo de dispositivos de red de A10, ACI, Aruba, AVI, CheckPoint, CloudEngine, CNOS, Eos, F5, Fortios, NetScaler…
- **Notification**
  - Mensajes en diferentes servicios como IRC, Jabber, Emails, Office365, Telegram, Twilio, etc., incluyendo **síntesis de voz**
- **Packaging**
  - Gestión de paquetes y librerías de Ruby, Python, Perl, Maven, Node .js, Apt, flatpak, homebrew, Macports, snap, yum,…
- **Remote Management**
  - Sistemas de gestión remota de equipos como Cobbler, CPM, DellEMC, Foreman, ILO de HP, IMC, OneView, RedFish, UCS,…

Versatiliidad de Ansible y sus Módulos

José Manuel
Redondo López

- **Source Control**
  - Módulos para gestión de GitHub y BitBucket
- **Storage**
  - Módulos para gestionar sistemas de almacenamiento tipo GlusterFS, EMC VNX, InfiniDAT, NetApp, PureStorage, Vexata y ZFS
- **System**
  - Gestión de distintos aspectos de SSOO (FreeBSD, Solaris…) y sus servicios (SELinux, UFW, usuarios, comandos típicos, …)
- **Utilities**
  - Son módulos que automatizan la gestión y ejecución de conceptos de Ansible (playbooks, tasks,…) vistos más adelante
- **Web Infrastructure**
  - Manejo de servidores Apache2, Django, JBoss, Nginx, Jira, Jenkins, Ansible Tower, Sophos UTM…
- **Windows**
  - Manejo general del sistema operativo Windows y sus funciones, comandos, Powershell, ficheros Chocolatey, registro, IIS, …

Ejemplo de ejecución de un módulo: ping

José Manuel
Redondo López

- **Como primer ejemplo vamos a usar un módulo para comprobar conectividad**
  - Ansible asume que se tiene acceso SSH a cualquier máquina del inventario
  - Intentará conectarse a la máquina remota como el usuario con el que se ejecuta
  - De ahí que sea tan importante el proceso conexión automática vía certificados
  - Así los comandos se ejecutarán sin problemas
- **En la imagen ejecutamos el comando anterior sobre el grupo `nginx_web`:**
  - `ansible nginx_web –m ping`
  - Vemos que uno funciona, pero el otro falla
  - Es necesario arreglar la conectividad desde el host de Ansible **a todos los host del inventario**
  - El módulo `ping` permite probarla cuando queramos

```
operario@server1804:~$ ansible nginx_web -m ping
192.168.1.200 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
192.168.1.201 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: ssh: connect to host 192.168.
1.201 port 22: No route to host",
    "unreachable": true
}
operario@server1804:~$
```

Ejemplo de ejecución de un módulo: ping

José Manuel
Redondo López

- Arreglada la conectividad, ya estamos listos para manejarlas siguiendo nuestras órdenes
- Hay que tener en cuenta que gracias a Ansible y las etiquetas del inventario
  - Podremos manipular de la misma forma **varias máquinas al mismo tiempo**
  - Y de esta forma crear configuraciones homogéneas en todas con una sola operación
  - **¡Nos puede ahorrar mucho tiempo!**

```
operario@server1804:~$ ansible nginx_web -m ping
192.168.2.200 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
192.168.2.201 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Ejemplo de ejecución de un módulo: ping

José Manuel
Redondo López

- **En resumen, esta es una lista de parámetros comunes**
  - `-i <ruta de un fichero>`: Establecer el archivo de inventario
    - Que por defecto está en `/etc/ansible/hosts`
    - Si no se especifica, se usará el de por defecto
  - `<nombre de etiqueta, ej.: local, nginx_server>`: usa Ansible contra los servidores definidos bajo las etiquetas especificadas en el archivo de inventario de hosts
    - `all` ejecutaría Ansible en todos los servidores definidos en ese archivo
  - **`-m <nombre de módulo>`: Ejecuta el módulo indicado y devuelve sus resultados**
    - El módulo "`ping`" ejecuta el comando `ping`
    - Veremos más módulos a continuación
  - `-c local` o `–connection=local`: Ejecuta comandos en el servidor local, no a través de SSH

Ejemplo de ejecución de un módulo: shell

José Manuel
Redondo López

- El módulo `shell` nos permite ejecutar comandos `bash` de la forma que estamos acostumbrados
  - ¡Puede usarse para **traducir directamente** nuestros scripts bash a Ansible!
  - La principal ventaja es que lo haría en todos los servidores del inventory
- Ej.: Usar este módulo para instalar `nginx` en nuestros servidores
  - `ansible -b --become-user=root –become-method=sudo --ask-become-pass nginx_web -m shell -a 'apt-get install –y nginx’`
- Un parámetro extra `–vvvv` le dice a Ansible que muestre información muy detallada de lo que va haciendo
  - Se usa para depuración

Ejemplo de ejecución de un módulo: shell

José Manuel
Redondo López

- **Veamos otra serie de parámetros comunes**
  - `-b` ("become"): Indica que va a ejecutar el comando **con otra identidad de usuario**
  - `--become-user=root`: Ejecuta los comandos como usuario `root`
    - Pero podemos definir **cualquier usuario existente** en las máquinas de destino
  - `become-method=sudo`: Método por el cual se va a adquirir la identidad de usuario
    - Indicamos `sudo`, pero existen más métodos
    - https://docs.ansible.com/ansible/latest/user_guide/become.html
  - `--ask-become-pass`: Le decimos a Ansible que nos pregunte por la clave a usar con `sudo`
    - Que tendrá que ser **la misma para todos los servidores**
  - `-a`: Usado para pasar cualquier argumento al módulo definido con `-m`
    - Los argumentos que admite cada módulo **dependen de su documentación**
    - Es **MUY IMPORTANTE** que el comando usado no espere una respuesta del usuario
      - De ahí que usemos `-y` con el `apt-get`
    - De esta forma logramos total flexibilidad a la hora de comunicarnos con los módulos

Documentación de módulos

José Manuel
Redondo López

- Todo módulo de **Ansible** está completamente documentado
- La imagen muestra la documentación del módulo `shell`
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/shell_module.html
  - En ella podemos encontrar sus argumentos
  - Y también para que sirve cada uno

Synopsis

- The `shell` module takes the command name followed by a list of space-delimited arguments.
- Either a free form command or `cmd` parameter is required, see the examples.
- It is almost exactly like the command module but runs the command through a shell (`/bin/sh`) on the remote node.
- For Windows targets, use the win_shell module instead.

Parameters

Parameter	Choices/Defaults	Comments
chdir path		Change into this directory before running the command.
cmd string		The command to run followed by optional arguments.
creates path		A filename, when it already exists, this step **will not** be run.
executable path		Change the shell used to execute the command. This expects an absolute path to the executable.
free_form string		The shell module takes a free form command to run, as a string. There is no actual parameter named 'free form'. See the examples on how to use this module.
removes path		A filename, when it does not exist, this step **will not** be run.
stdin string *added in 2.4*		Set the stdin of the command directly to the specified value.
stdin_add_newline boolean *added in 2.8*	Choices: • no • **yes** ←	Whether to append a newline to stdin data.
warn boolean	Choices: • no • **yes** ←	Whether to enable task warnings.

EJEMPLO DE EJECUCIÓN DE UN MÓDULO: APT

José Manuel
Redondo López

- El módulo `shell` es extremadamente flexible
- Pero también "especial", porque no sigue la tónica general del resto de módulos: **no puede garantizar la idempotencia**
  - Es decir, ejecutar las mismas tareas múltiples veces sin afectar al resultado final
  - En otras palabras, un módulo Ansible normalmente asegura que si una operación ya se ha hecho anteriormente **NO** se vuelva a hacer
- Por ese motivo, para instalar software en Debian/Ubuntu es más adecuado el módulo `apt`
  - Ejecutará el mismo comando, pero garantizará la idempotencia gracias al uso de **facts**
  - **Si nginx ya está instalado, no lo volverá a hacer**
  - `ansible -b -m apt -a 'name=nginx state=present update_cache=true'`

Documentación de módulos: APT

José Manuel
Redondo López

- Cada módulo tiene también su propia serie de parámetros
- O al comando que representan
  - Ej.: `update-cache`
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/apt_module.html
  - Como se puede ver, están relacionados con el gestor de paquetes `apt`
- **Es la tónica general para el resto de los módulos**
  - Cada uno tiene los suyos, relacionados con su función

Synopsis

- Manages *apt* packages (such as for Debian/Ubuntu).

Requirements

The below requirements are needed on the host that executes this module.

- python-apt (python 2)
- python3-apt (python 3)
- aptitude (before 2.4)

Parameters

Parameter	Choices/Defaults	Comments
allow_unauthenticated boolean	Choices: • no ← • yes	Ignore if packages cannot be authenticated. This is useful for bootstrapping environments that manage their own apt-key setup. `allow_unauthenticated` is only supported with state: *install/present*
autoclean boolean *added in 2.4*	Choices: • no ← • yes	If `yes`, cleans the local repository of retrieved package files that can no longer be downloaded.
autoremove boolean	Choices: • no ← • yes	If `yes`, remove unused dependency packages for all module states except *build-dep*. It can also be used as the only option. Previous to version 2.4, autoclean was also an alias for autoremove, now it is its own separate command. See documentation for further information.
cache_valid_time -	Default: 0	Update the apt cache if its older than the *cache_valid_time*. This option is set in seconds. As of Ansible 2.4, if explicitly set, this sets *update_cache=yes*.
deb -		Path to a .deb package on the remote machine. If :// in the path, ansible will attempt to download deb before installing. (Version added 2.1) Requires the `xz-utils` package to extract the control file of the deb package to install.
default_release -		Corresponds to the `-t` option for *apt* and sets pin priorities
dpkg_options -	Default: "force-confdef,force-confold"	Add dpkg options to apt command. Defaults to '-o "Dpkg::Options::=--force-confdef" -o "Dpkg::Options::=--force-confold"' Options should be supplied as comma separated list
force boolean	Choices: • no ← • yes	Corresponds to the `--force-yes` to *apt-get* and implies `allow_unauthenticated: yes` This option will disable checking both the packages' signatures and the certificates of the web servers they are downloaded from. This option *is not* the equivalent of passing the `-f` flag to *apt-get* on the command line **This is a destructive operation with the potential to destroy your system, and it should almost never be used.** Please also see `man apt-get` for more information.
force_apt_get boolean *added in 2.4*	Choices: • no ← • yes	Force usage of apt-get instead of aptitude
install_recommends boolean	Choices: • no	Corresponds to the `--no-install-recommends` option for *apt*. `yes` installs recommended packages. `no` does not install recommended packages. By default, Ansible will use never installed.

Ejecución de un módulo idempotente

José Manuel
Redondo López

- Este ejemplo usa el módulo `apt` para actualizar la caché del repositorio e instalar **Nginx**
  - El resultado de ejecutar la tarea fue "`changed`": `false`
  - No hubo cambios: ya se había instalado **Nginx** antes usando el módulo `shell`
- Hemos visto como ejecutar las tareas a través de módulos individualmente
- Pero para sacarle más partido, se pueden mover las tareas a **Playbooks**
  - ¡Pueden ejecutar y coordinar varias tareas!
  - Es la siguiente sección ☺

```
operario@server1804:~$ ansible -b --become-user=root --become-method=sudo --ask-become-pass nginx_web -m apt -a
 'name=nginx state=present update_cache=true'
BECOME password:
 [WARNING]: Could not find aptitude. Using apt-get instead

192.168.2.200 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1562928592,
    "cache_updated": true,
    "changed": false
}
192.168.2.201 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1562928593,
    "cache_updated": true,
    "changed": false
}
operario@server1804:~$
```

Ejecución de un módulo que admite listas de elementos

José Manuel
Redondo López

- Muchos módulos tienen opciones que admiten varios elementos
- Para todos esos casos es necesario recurrir a la **sintaxis de lista** de YAML
- Básicamente consiste en listar los elementos con un **–** delante
- ¡Siempre manteniendo una indentación coherente!
  - **Recuerda**: Hay Python por debajo...
- Se muestra un ejemplo con `apt`, pero hay muchos otros

```
- name: “Instalar software base"
    apt:
      state: present
      name:
        - acl
        - git
        - curl
        - unzip
        - sendmail
        - apache2
```

Otros módulos interesantes

José Manuel
Redondo López

- `lineinfile`: Que permite asegurarnos de que hay una línea concreta en un fichero
  - O reemplazarla por otra usando expresiones regulares
  - Especialmente útil para cambiar ficheros de configuración de programas
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/lineinfile_module.html
- `block`: permite organizar la ejecución de tareas en bloques
  - Además, permite manejar errores en alguna de las tareas al final del bloque, como excepciones (hablaremos de errores posteriormente)
  - https://docs.ansible.com/ansible/latest/user_guide/playbooks_blocks.html
- `set_fact`: Permite cambiar el valor de una fact durante la ejecución de una tarea de Ansible
  - Como escribir en una variable un valor en función de lo que haya pasado en la ejecución
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/set_fact_module.html

¿Crees que lo has entendido todo? ¡Autoevalúate!

José Manuel
Redondo López

?

Facts

- *¿Entiendes que es una **fact** de una máquina remota y el tipo de información que te dan?*
- *¿Comprendes que gracias a esa información puedes tomar decisiones de administración mucho más potentes dentro del código **Ansible** que diseñes?*

Módulos

- *¿Comprendes que todo lo que **Ansible** hace es gracias a sus módulos?*
- *¿Entiendes que los módulos son bibliotecas reutilizables con propósitos definidos, que al estar perfectamente documentados y probados pueden facilitarte mucho la administración?*
- *¿Comprendes que los módulos están agrupados por función para facilitar su búsqueda?*
- *¿Has comprendido como comprobar si una máquina remota puede ser controlada correctamente por **Ansible** con el módulo **ping**?*
- *¿Entiendes que el módulo **shell** es darte la capacidad de ejecutar en remoto cualquier orden de línea de comandos, y que con eso puedes usar cualquier script que ya tengas de antes?*
- *¿Comprendes por qué el módulo **shell** no es idempotente entonces?*
- *¿Has entendido la estructura de la documentación de cualquier módulo **Ansible**?*
- *¿Comprendes cómo trabajar de forma general con un módulo **Ansible** cualquiera?*
- *¿Entiendes qué significa que un módulo sea idempotente?*

Playbooks, Tasks y Handlers

Usando Ansible para manejar nuestros servidores

¿QUÉ VAMOS A VER EN ESTE APARTADO?

José Manuel
Redondo López

- **En este apartado vamos a hablar de los conceptos principales de Ansible que nos quedan: Playbooks, Tasks, Handlers, Roles y Templates**
  - Como los playbooks son los **programas Ansible** que nos permiten automatizar la administración de máquinas
  - La **división de playbooks** en unidades llamadas tasks
  - Como los handlers no son más que **tasks especiales** que se activan en momentos concretos
  - Como los roles son una forma de **encapsulamiento de playbooks** complejos
  - El uso de templates para ahorrar **escribir código** Ansible

PLAYBOOKS Y TASKS

José Manuel
Redondo López

- **Los playbooks permiten ejecutar varias tareas (Tasks) y tienen funciones más avanzadas**
  - Se dividen en acciones (las Tasks) en forma de lista YAML
  - Cada task empieza por – y name
  - **Pueden contener muchas task** (varios –name)
- **Se define en lenguaje YAML (.yml)**
  - Los ficheros YAML **solo usan espacios** para anidar elementos **NUNCA tabuladores**
  - **Es muy estricto** con los espacios y el anidamiento de sus elementos, al estar escrito en Python
    - Elementos al mismo nivel **deben estar alineados igual**
    - Debemos **respetar una estructura y sintaxis**
    - https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html
- **Este Playbook equivale al comando anterior**

¡Un editor como Visual Studio Code con los *plugins* adecuados puede ayudar mucho a desarrollar ficheros Ansible! **(y detectar caracteres "raros" al copiar código)**

```yaml
---
hosts: nginx_web
# No, son hosts remotos!
#  connection: local
  become: yes
  become_user: root
  become_method: sudo
  tasks:
   - name: Instalar Nginx base
     apt: # Módulo y sus parámetros
       name: nginx
       state: present
       update_cache: true
```

AnsibleTask.yml

PLAYBOOKS Y TASKS

José Manuel
Redondo López

- Las tareas **pueden ser asíncronas**, pero por defecto no lo son
- Vemos que hay una etiqueta equivalente a cada parámetro usado en el comando anterior
- Para ejecutar el fichero **YAML** del **Playbook** necesitamos usar el comando `ansible-playbook`
  - `ansible-playbook nginx.yml`
- Muestra una traza de las tareas ejecutadas y su resultado
  - Nuevamente no hay cambios porque Nginx ya estaba instalado de antes (**idempotencia**)

```
operario@server1804:~$ ansible-playbook --ask-become-pass nginx.html
BECOME password:

PLAY [nginx_web] ***************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [192.168.2.200]
ok: [192.168.2.201]

TASK [Instalar Nginx base] *****************************************************************************************
 [WARNING]: Could not find aptitude. Using apt-get instead

ok: [192.168.2.201]
ok: [192.168.2.200]

PLAY RECAP *********************************************************************************************************
192.168.2.200              : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignore
d=0
192.168.2.201              : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignore
d=0
```

HANDLERS

José Manuel
Redondo López

- Un **handler** puede hacer las mismas cosas que una tarea
- Pero sólo se ejecutarán cuando una los llame
  - Son útiles para acciones "secundarias“, necesarias **después de ejecutar una tarea concreta**
  - Es algo similar a un evento que ocurre cuando una tarea termina, por ejemplo
- Ejemplos típicos son
  - Iniciar un nuevo servicio después de la instalación
  - Volver a cargar un servicio tras un cambio de configuración
- Permiten una **ejecución sincronizada** basada en cuándo acaban ciertas cosas para iniciar otras

```yaml
---
- hosts: nginx_web
  become: yes
  become_user: root
  become_method: sudo
  tasks:
    - name: Instalar Nginx base
      apt: # Módulo
        name: nginx
        state: present
        update_cache: true
      notify:
        - Arranca Nginx
  handlers:
    - name: Arranca Nginx
      service: # Otro módulo
        name: nginx
        state: started
```

`AnsibleHandlers.yml`

HANDLERS

José Manuel
Redondo López

- **Hemos añadido una directiva `notify` a la tarea de instalación**
  - Esto notificaría a cualquier `handler` denominado `"Arranca Nginx"` después de ejecutar la tarea
  - Ahora podemos crear el `handler` llamado `"Arranca Nginx"` para que se llame
  - Este `handler` usa el **módulo `service`**
    - Que permite iniciar, detener, reiniciar y volver a cargar servicios del sistema
- **En este caso, le decimos a Ansible que queremos que se inicie Nginx cuando se termine de instalar**
  - Pero lo que hacemos es **definir el estado** en el que queremos el servicio
  - A través de las **facts**, Ansible decidirá si se necesita (o no) un cambio de estado en el servicio
    - Por supuesto, en función del que tenga actualmente
    - Pero eso lo hace él, ¡tú no tienes que comprobar nada de esto!

HANDLERS

José Manuel
Redondo López

- Ahora ejecutamos de nuevo el **Playbook** con el **handler** añadido

```
operario@server1804:~$ ansible-playbook --ask-become-pass nginx2.html
BECOME password:

PLAY [nginx_web] ***************************************************************************************

TASK [Gathering Facts] *********************************************************************************
ok: [192.168.2.200]
ok: [192.168.2.201]

TASK [Instalar Nginx base] *****************************************************************************
 [WARNING]: Could not find aptitude. Using apt-get instead

ok: [192.168.2.201]
ok: [192.168.2.200]

PLAY RECAP *********************************************************************************************
192.168.2.200              : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignore
d=0
192.168.2.201              : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignore
d=0

operario@server1804:~$
```

Uso de Playbooks

José Manuel
Redondo López

- La salida de la ejecución es igual a la anterior
- Se debe a que las directivas `notify` **solo se ejecutan si se ejecuta la tarea asociada**
  - Al tener Nginx ya instalado, la tarea "Instalar Nginx" no se ejecutaría y, por tanto, **no se llama** a su `notify` asociado
  - ¡Máxima eficiencia!
- Es importante darse cuenta de cómo podemos usar Playbooks para ejecutar cosas más avanzadas
  - Múltiples tareas
  - Añadir variables
  - Definir parámetros de configuración
  - Incluir otros playbooks
  - …
- Veamos un ejemplo

Uso de Tasks "en orden": Register y When

José Manuel
Redondo López

- Ahora queremos tres tasks
  - **Incluir el repo oficial de Nginx**
    - Añade el PPA estable de Nginx para usar su última versión estable; **módulo** `apt_repository`
  - **Instalar Nginx**
    - Luego, instala Nginx utilizando el **módulo** `apt`
  - **Crear el alojamiento raíz de las webs**
    - Crea un directorio raíz web como último paso del Playbook con el **módulo** `file`
- Usaremos dos directivas nuevas que nos permiten ejecutar tareas en un orden determinado si han pasado cosas antes (el equivalente a un `if`)
  - `register`: La tarea "Incluir el repo oficial de Nginx" registra la variable `ppastable`
  - `when`: Solo ejecuta la tarea asociada **cuando alguna otra tarea ha registrado la variable** indicada
    - En este ejemplo, la tarea "Instalar Nginx" solo se ejecuta cuando la tarea que registra la variable `ppastable` finaliza correctamente
    - Esto permite que ciertas tareas de Ansible se ejecuten **condicionalmente y "en orden"**

Uso de Playbooks

José Manuel
Redondo López

```yaml
---
- hosts: nginx_web
  become: yes
  become_user: root
  become_method: sudo
  vars:
    - webroot: /var/www/html/pruebas_ansible
  tasks:
    - name: Incluir el repo oficial de Nginx # Primera tarea
      apt_repository: # módulo
        repo: ppa:nginx/stable
        state: present
      register: ppastable
    - name: Instalar Nginx # Segunda tarea
      apt: # módulo
        pkg: nginx
        state: present
        update_cache: true
      when: ppastable is success
      notify:
        - Arranca Nginx
    - name: Crear el alojamiento raíz de las web # Tercera tarea
      file: # módulo
        path: '{{ webroot }}'
        mode: 775
        state: directory
        owner: www-data
        group: www-data
      notify:
        - Recarga Nginx
  handlers:
    - name: Arrranca Nginx
      service: # módulo
        name: nginx
        state: started

    - name: Recarga Nginx
      service: # módulo
        name: nginx
        state: reloaded
```

AnsibleUsoPlaybooks.yml

Usos de Register y When

José Manuel
Redondo López

- Este ejemplo particular se ha hecho para ilustrar la funcionalidad de secuenciación de tareas
- En un caso real, **si se produce un error** al agregar el repositorio, **Ansible detendrá la ejecución** del **Playbook** e informará del error
- También pueden registrar los resultados de las acciones de un módulo
  - Y usar la variable definida en `register` para realizar acciones condicionalmente con `when`, según los valores de las variables registradas
  - Por ejemplo, si **se registra el resultado de la ejecución** del comando a través del módulo `shell`, se podrá acceder a la salida estándar de ese comando y usarla para tomar decisiones
  - Ejemplo: https://stackoverflow.com/questions/36059804/ansible-store-commands-stdout-in-new-variable

```
tasks:
      - shell: echo $(cat /etc/issue | awk {'print $7'})
        register: echo_content

      - shell: echo "It works"
        when: echo_content.stdout == "12"
        register: out
      - debug: var=out.stdout_lines
```

Guarda el resultado de la ejecución en `echo_content` y luego usa su valor para ejecutar (o no) una tarea que imprime el resultado de otra ejecución. Usar `shell` es adecuado para obtener información del entorno (la no idempotencia aquí da igual)

Uso de Playbooks: Variables

José Manuel
Redondo López

```yaml
---
- hosts: nginx_web
  become: yes
  become_user: root
  become_method: sudo
  vars:
    - webroot: /var/www/html/pruebas_ansible
  tasks:
    - name: Incluir el repo oficial de Nginx # Primera tarea
      apt_repository: # módulo
        repo: ppa:nginx/stable
        state: present
      register: ppastable
    - name: Instalar Nginx # Segunda tarea
      apt: # módulo
        pkg: nginx
        state: present
        update_cache: true
      when: ppastable is success
      notify:
        - Arranca Nginx
    - name: Crear el alojamiento raíz de las web # Tercera tarea
      file: # módulo
        path: '{{ webroot }}'
        mode: 775
        state: directory
        owner: www-data
        group: www-data
      notify:
        - Recarga Nginx
  handlers:
    - name: Arrranca Nginx
      service: # módulo
        name: nginx
        state: started

    - name: Recarga Nginx
      service: # módulo
        name: nginx
        state: reloaded
```

AnsibleUsoPlaybooks.yml

Uso de Playbooks: Variables

José Manuel
Redondo López

- **El ejemplo anterior también usa una variable en una sección `var` llamada `webroot`**
  - Se usa luego como argumento del módulo `file`, que **crea el directorio** definido por dicha variable
- **El uso de variables usa la sintaxis de plantillas Jinja2**
  - https://docs.ansible.com/ansible/latest/user_guide/playbooks_templating.html
  - Básicamente usa **dobles corchetes** para referirse a la variable, **rodeados de comillas** simples o dobles
    - Ej.: `'{{<nombre de variable>}}'`
  - Las variables pueden definirse sueltas, pero también
    - En forma **de lista**
      - Elementos precedidos de `-` y con un nombre
    - En forma **de diccionario**
      - Idéntico al anterior, pero en forma de `<nombre>:<valor>`
      - Y referenciándose como `<nombre del diccionario>['<nombre>']` o `<nombre del diccionario>.<nombre>`
    - Más información: https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html

REGISTER Y VARIABLES

José Manuel
Redondo López

- **Como ya te habrás percatado, `register` crea variables a partir del resultado de una tarea**
  - Esas variables se pueden usar posteriormente en el playbook
  - Son variables adicionales a las que se pueden crear en otras partes del playbook
- **Estas variables quedan definidas dentro de la estructura del elemento YAML que las define**
  - Para acceder a su valor hay que usar la sintaxis de . (`{{ a.b.c.d }}`)
  - O de diccionario `{{ a["b"]["c"]["d"] }}`
- **Es muy típico en las facts obtenidas de los sistemas remotos manejados**
  - Ej.: `{{ ansible_facts.eth0.ipv4.address }}`

Uso de Playbooks

José Manuel
Redondo López

```
operario@server1804:~$ ansible-playbook --ask-become-pass nginx3.html
BECOME password:

PLAY [nginx_web] ***************************************************************************************

TASK [Gathering Facts] *********************************************************************************
ok: [192.168.2.200]
ok: [192.168.2.201]

TASK [Incluir el repo oficial de Nginx] ****************************************************************
changed: [192.168.2.200]
changed: [192.168.2.201]

TASK [Instalar Nginx] **********************************************************************************
 [WARNING]: Could not find aptitude. Using apt-get instead
ok: [192.168.2.200]
ok: [192.168.2.201]

TASK [Crear el alojamiento raiz de las web] ************************************************************
changed: [192.168.2.200]
changed: [192.168.2.201]

RUNNING HANDLER [Recarga Nginx] ************************************************************************
changed: [192.168.2.201]
changed: [192.168.2.200]

PLAY RECAP *********************************************************************************************
192.168.2.200              : ok=5    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignore
d=0
192.168.2.201              : ok=5    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignore
d=0
```

FACTS COMO VARIABLES

José Manuel
Redondo López

- **Cada uno de los facts extraídos de las máquinas controladas puede usarse como una variable en un Playbook, con la misma sintaxis**
  - Por ejemplo, si se quieren utilizar un nº de `worker_processes` en Nginx igual al nº de cores de la CPU se puede usar la variable `{{ ansible_processor_cores }};`
  - Si hay varias CPU: `{{ ansible_processor_cores * ansible_processor_count }};`
  - **Recuerda**: Todos los Facts de Ansible empiezan por `ansible_`
- **Cualquier variable (fact o no) puede manipularse por código**
  - Con filtros predefinidos de Jinja2
    - https://jinja.palletsprojects.com/en/3.1.x/templates/#builtin-filters
  - Con filtros específicos de Ansible
    - https://docs.ansible.com/ansible/latest/user_guide/playbooks_filters.html#playbooks-filters
  - Esto permite la manipulación avanzada de las mismas
  - Aunque es un aspecto que se deja para que lo investiguéis si os hace falta

GESTIÓN DE ERRORES DE EJECUCIÓN DE PLAYBOOKS

José Manuel
Redondo López

- En caso de error de un comando o módulo, por defecto Ansible para de ejecutarse en un host y "salta" al siguiente
- No obstante, ese comportamiento puede no ser adecuado en algunos casos, y puede cambiarse de varias formas
  - https://docs.ansible.com/ansible/latest/user_guide/playbooks_error_handling.html
  - Si queremos **ignorar los errores** al ejecutar una tarea, le añadimos `ignore_errors: yes`
  - Si queremos que **se ignore si un host no está disponible** al ejecutarla: `ignore_unreachable: yes`
    - Podemos hacer lo mismo a nivel de playbook
- También podemos definir qué entendemos como error a través de `failed_when` y analizando una variable que registremos (`register`) con el resultado de la tarea

```
- name: Comprobar si un fichero existe y dar fallo si no
  ansible.builtin.command: ls /tmp/ficheruco
  register: result
  failed_when:
     - result.rc == 0
     - '"No such" not in result.stdout'
```

¿Y SI LA TAREA ME HACE PREGUNTAS?

José Manuel
Redondo López

- Como dijimos, debemos evitar que una tarea haga preguntas al usuario
  - Si las hacen, **la ejecución se interrumpirá**
- Pero podemos saltarnos esa limitación con el módulo `expect`:
  - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/expect_module.html
  - ¡Podemos indicar respuestas a nuestro gusto para cualquier pregunta que el script haga!

```
- name: Ejecutar un script que hace preguntas
  expect:
    command: "/bin/bash /install.sh"
    timeout: 300
    responses:
      en/es: "en"
      Press ENTER: ""
      What kind of installation do you want: "{{ variable_con_respuesta }}"
      Choose where to install: "{{ variable_con_path }}"
      Do you want e-mail notification: "{{ variable_con_email }}"
```

Elementos avanzados de las Tasks

José Manuel
Redondo López

- **Las tareas admiten además otros elementos**
  - **Bucles**: formas de definir una lista de variables de manera que una misma task se ejecute sobre cada elemento de dicha lista
    - Pueden además ser pares, tripletas, etc.
    - Pueden llegar a ser muy complejos: https://docs.ansible.com/ansible/latest/user_guide/playbooks_loops.html
  - **Tareas pre y post**: hay un par de tareas especiales: `pre_tasks` y `post_tasks`
    - Como su nombre indica, las `pre_tasks` de un playbook se ejecutarán antes de cualquier otra task
    - Esto incluye a los roles, que veremos más adelante
    - Las `post_tasks` **son lo contrario**
      - Se ejecutan después de todas las tareas, incluyendo los handlers definidos por otras tareas

```
pre_tasks:
   - name: Update apt cache if needed.
     apt: update_cache=yes cache_valid_time=3600
```

Recapitulando...

José Manuel
Redondo López

- Hasta ahora hemos usado **Ansible** de varias formas
  - Ejecutando comandos aislados
  - Utilizando módulos
  - Organizando tareas en **Playbooks**
- Cada uno de ellos nos permitía hacer cosas de complejidad mayor y lograr más control y organización
- Pero podemos incrementar aún más el nivel de organización y la complejidad de lo que se ejecuta gracias a los **Roles** y **Templates**
- Que será lo que veamos en la siguiente sección

¿Crees que lo has entendido todo? ¡Autoevalúate!

José Manuel
Redondo López

- *¿Entiendes lo que es un **playbook** y como modela una lista de tareas?*
- *¿Crees que has entendido la estructura de un **playbook** básica y su funcionamiento síncrono (el orden de las tareas importa)?*
- *¿Has entendido el propósito general de un **handler** y cuándo y cómo se ejecuta?*
- *¿Entiendes cómo afecta la idempotencia de tareas a la ejecución de **handlers**?*
- *¿Comprendes como con **register** y **when** puedes modelar condicionales en **Ansible**?*
- *¿Comprendes cómo se usa y se declara una variable en **Ansible**?*
- *¿Tienes claro que las **facts** se pueden usar como variables en un **playbook**?*
- *¿Entiendes la gestión de errores por defecto de **Ansible**, y cómo cambiarla?*
- *¿Comprendes cómo controlar la ejecución de **scripts** que hagan preguntas?*
- *¿Tienes claro que **Ansible** permite bucles y ejecutar tareas pre y post ejecución?*

Roles y Templates

Mejorando la organización de nuestros elementos

Roles

José Manuel
Redondo López

- **A medida que vamos haciendo más cosas la organización de los Playbook se vuelve más compleja**
  - Piensa en que es como en programación clásica: cuando un programa crece, dejarlo todo en el mismo fichero **lo hace mucho menos mantenible**
  - Entonces lo intentas partir **en módulos**, uno para cada cosa
  - Verás que esto es lo mismo, pero **con una estructura más rígida**
- **Los roles sirven para organizar tareas de Ansible que estén relacionadas**
  - Y encapsular los datos necesarios para realizarlas
  - Un rol permite hacerlo con **una estructura coherente y muy organizada**
  - Tendremos las variables, ficheros y otros elementos **más fácilmente localizables**

Roles

José Manuel
Redondo López

- Los roles tienen una estructura de directorios como la adjunta
- En cada directorio Ansible **busca y lee automáticamente archivos llamados** `main.yml`
  - Salvo en `files` y `templates`, que contienen otro tipo de ficheros
- La idea es separar un **Playbook** como el anterior en partes
  - Y colocar cada parte dentro del directorio correspondiente
- Así todo el conjunto quedará más limpio y organizado

```
<carpeta raíz de todos los roles>
  <nombre del rol>
    - files
    - handlers
    - meta
    - templates
    - tasks
    - vars
    - defaults
    - tests
```

Creación de un rol

José Manuel
Redondo López

- **El comando `ansible-galaxy` se puede usar para crear un nuevo rol localmente**
  - También sirve para guardar roles en el **registro público de Ansible** que veremos más adelante
- **Para ello**
  - Creamos un directorio para todos los roles y accedemos a él
    - `mkdir roles`
    - `cd roles`
  - Creamos un nuevo rol al que llamaremos `nginx`
    - `ansible-galaxy init nginx`

Ejemplo de ejecutar: `ansible-galaxy init mongodb`

Creación de un rol

José Manuel
Redondo López

- El nombre de directorio `roles` es el que **Ansible** usa para encontrar **Playbooks** y **debería tener siempre este nombre**
- `ansible-galaxy init nginx` ejecutado dentro del directorio `roles`, creará los directorios/archivos necesarios para crear un nuevo rol
- Las siguientes transparencias enumeran cada parte de un rol
  - Indicando qué va en cada una de ellas

```
operario@server1804:~$ mkdir ansible-example
operario@server1804:~$ cd ansible-example/
operario@server1804:~/ansible-example$ mkdir roles
operario@server1804:~/ansible-example$ cd roles/
operario@server1804:~/ansible-example/roles$ ansible-galaxy init nginx
- nginx was created successfully
operario@server1804:~/ansible-example/roles$ ls
nginx
operario@server1804:~/ansible-example/roles$ cd nginx
operario@server1804:~/ansible-example/roles/nginx$ ls
defaults  files  handlers  meta  README.md  tasks  templates  tests  vars
operario@server1804:~/ansible-example/roles/nginx$
```

Rol: Files

José Manuel
Redondo López

- Dentro de `files` podemos añadir **archivos a copiar en nuestros servidores**
- Es típico añadir a esta carpeta la configuración del servidor web
  - O de otros programas, o incluso webs a servir…
  - Ej.: Una configuración propia o "prefabricada"
    - Como la que se puede encontrar aquí
      - https://github.com/h5bp/server-configs-nginx/tree/master/h5bp
    - Para ello descargamos el contenido del repositorio `git` a un directorio
    - Y copiamos el subdirectorio `h5bp` al directorio `files` del rol
      - `git clone https://github.com/h5bp/server-configs-nginx.git`

```
operario@server1804:~/temp$ git clone https://github.com/h5bp/server-configs-ngi
nx.git
Cloning into 'server-configs-nginx'...
remote: Enumerating objects: 134, done.
remote: Counting objects: 100% (134/134), done.
remote: Compressing objects: 100% (92/92), done.
remote: Total 1739 (delta 64), reused 76 (delta 38), pack-reused 1605
Receiving objects: 100% (1739/1739), 364.00 KiB | 1.19 MiB/s, done.
Resolving deltas: 100% (919/919), done.
operario@server1804:~/temp$ ls
server-configs-nginx
operario@server1804:~/temp$ cd server-configs-nginx/
operario@server1804:~/temp/server-configs-nginx$ ls
CHANGELOG.md  h5bp         mime.types  README.md
conf.d        LICENSE.txt  nginx.conf  test
operario@server1804:~/temp/server-configs-nginx$ cp h5bp/ /home/operario/ansible
-example/roles/nginx/files/
cp: -r not specified; omitting directory 'h5bp/'
operario@server1804:~/temp/server-configs-nginx$ cp -r h5bp/ /home/operario/ansi
ble-example/roles/nginx/files/
operario@server1804:~/temp/server-configs-nginx$
```

```
operario@server1804:~/ansible-example/roles/nginx/files$ ls
h5bp
operario@server1804:~/ansible-example/roles/nginx/files$ cd h5bp/
operario@server1804:~/ansible-example/roles/nginx/files/h5bp$ ls
basic.conf    errors            location    security  web_performance
cross-origin  internet_explorer  media_types  ssl
```

ROL: FILES

José Manuel
Redondo López

- **Descargarse esta configuración segura y probada está bien**
  - Pero, en este caso concreto, para que todos sus elementos funcionen **hacen falta módulos de Nginx adicionales**
    - Se hablo de Nginx y módulos en el tema 1
  - En este caso tenemos una instalación por defecto, por lo que nos faltan muchos de ellos
  - Por tanto, tenemos que desactivar parte de la configuración para trabajar con un Nginx básico
  - El fichero `basic.conf` de la configuración de ejemplo descargada debería quedar como el de la imagen para evitar errores
- **Esto no es lo que se hace habitualmente**
  - Querrás aprovecharte del material instalando **todas las dependencias** que necesite

```
# https://github.com/h5bp/server-configs-nginx

#include h5bp/internet_explorer/x-ua-compatible.conf;
#include h5bp/security/referrer-policy.conf;
include h5bp/security/x-content-type-options.conf;
#include h5bp/security/x-frame-options.conf;
#include h5bp/security/x-xss-protection.conf;
include h5bp/location/security_file_access.conf;
#include h5bp/cross-origin/requests.conf;
```

ROLES: HANDLERS

José Manuel
Redondo López

- Dentro del directorio `handlers`, podemos poner todos los **handlers** que teníamos en el **playbook** `nginx.yml`
- Para ello, dentro de `handlers/main.yml` ponemos el contenido que vemos a la derecha
- Podemos hacer referencia a ellos desde otra configuración YAML

```yaml
---
- name: Arranca Nginx
  service:
    name: nginx
    state: started

- name: Recarga Nginx
  service:
    name: nginx
    state: reloaded
```

Roles: Meta

José Manuel
Redondo López

- **Hoy en día un servidor web en producción DEBE servir todas sus webs bajo HTTPs**
- **Pero, en este ejemplo ¿Cómo hacemos que un rol instale un certificado SSL para habilitar esto en la máquina destino?**
  - Por suerte, existen roles disponibles que ya facilitan esto
  - Un ejemplo es: https://github.com/ome/ansible-role-ssl-certificate
  - Para ello nos descargamos el contenido
    - `git clone https://github.com/ome/ansible-role-ssl-certificate.git`
  - Creamos un nuevo rol `ssl-certificate` con `ansible-galaxy` como hemos visto
  - Copiamos el contenido de la carpeta descargada a la creada para el rol
- **Otro ejemplo es este: https://github.com/weareinteractive/ansible-openssl**
- **Pero, y ahora...¿Cómo juntamos esto con la instalación del servidor web anterior?**

ROLES: META

José Manuel
Redondo López

- **El archivo `main.yml` dentro del directorio `meta` contiene metadatos de rol, incluyendo dependencias**
  - Si un rol dependiera de **otro rol que ya hayamos instalado**, podríamos definirlo aquí
  - Por ejemplo, el rol `nginx` depende del rol `ssl-certificate`, que instala certificados SSL que acabamos de ver, y se indicaría así

```
---
dependencies:
  - { role: ssl-certificate }
```

  - Ahora antes de ejecutarse el rol "`nginx`" **se intentaría ejecutar primero el rol** "`ssl-certificate`"
- **Si no tenemos dependencias o metadatos, podemos no poner el archivo o definirlo así**

```
---
dependencies: []
```

Roles: Vars y Defaults

José Manuel
Redondo López

- El directorio `vars` contiene un archivo `main.yml` con las variables que usaremos en otras partes del rol
- Ejemplo de `vars/main.yml`:

```
---
domain: serversample.local
```

- Es común que las variables tengan **información confidencial**
  - En ese caso debemos usar el **Vault de Ansible**, que veremos más adelante
- El directorio `defaults` contiene el valor por defecto de las variables en un `main.yml`
  - Este valor **tiene la menor prioridad**: cualquier otra definición de la variable en otra parte lo sobrescribe
  - https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html#role-default-variables

ROLES: TEMPLATE

José Manuel
Redondo López

- **Los archivos de plantilla (template) pueden contener variables de plantilla**
  - Basadas en el motor de plantillas Jinja2 de Python que mencionamos anteriormente
  - El nombre de estos archivos es arbitrario, pero su extensión debe ser `.j2`
- **La siguiente transparencia muestra un ejemplo de configuración de unos bloques server de Nginx**
  - Utiliza las variables que hemos definido en el directorio `vars`
  - El archivo se puede llamar `templates/localhost.conf.j2`

ROLES: TEMPLATE

José Manuel
Redondo López

```
# Plantilla de configuración para nginx
templates/localhost.conf.j2

server {

    # Obliga a usar HTTPS con una redirección

    listen 80 default_server;

    server_name {{ domain }};

    return 301 https://$server_name$request_uri;

}

server {

  listen 443 ssl default_server;


  root /var/www/{{ domain }}/public;

  index index.html index.htm index.php;


  access_log /var/log/nginx/{{ domain }}.log;

  error_log  /var/log/nginx/{{ domain }}-error.log error;

    server_name {{ domain }};

    charset utf-8;

    include h5bp/basic.conf;

    # Estas 2 variables están definidas en el rol ssl-certificate

    # que nos descargamos de GitHub antes

    ssl_certificate          {{ ssl_certificate_public_path }};

    ssl_certificate_key      {{ ssl_certificate_key_path }};

    include h5bp/ssl/ssl_engine.conf;


    location = /favicon.ico {log_not_found off;access_log off; }

    location = /robots.txt  {log_not_found off;access_log off; }

}
```

localhost.conf.j2

ROLES: FACTS EN TEMPLATES

José Manuel
Redondo López

- Cada uno de los **facts** extraídos de las máquinas controladas también puede usarse como una variable en una template
- Por ejemplo, si se quieren utilizar un nº de `worker_processes` en **Nginx** igual al nº de cores de la CPU se puede hacer en una plantilla `.j2` de la siguiente manera:

```
user www-data;
worker_processes {{ ansible_processor_cores }};
…
```

- Y si el servidor tiene varias CPU

```
user www-data;
worker_processes {{ ansible_processor_cores * ansible_processor_count }};
…
```

- También pueden usarse para cargar ficheros con variables o tareas condicionalmente según el nombre que tengan (Ej.: para distintas plataformas)
  - Con las directivas `include_vars` e `include_tasks`

ROLES: TEMPLATE

José Manuel
Redondo López

- **Esta es una configuración Nginx automatizada típica para una aplicación web cualquiera a partir de una segura**
  - Si instalásemos más módulos de Nginx podríamos activar más partes de la configuración vista e ir completando la configuración del servidor
  - Una vez terminada, podríamos exportarla a otro servidor similar
- **Usa las siguientes variables del directorio `vars` de ambos roles, sin distinguir entre ellos (`nginx` y `ssl-certificate`, descargado de GitHub)**
  - `domain`
  - `ssl_certificate_public_path`
  - `ssl_certificate_key_path`
- **La integración entre roles es por tanto transparente**
- **Todos los elementos se coordinan en las tasks del rol**
  - Los roles necesarios **se pueden preinstalar automáticamente como prerrequisitos** de un rol dado

ROLES: TASKS

José Manuel
Redondo López

```yaml
---
- name: Incorporar el repo Nginx
  apt_repository:
    repo: ppa:nginx/stable
    state: present

- name: Instalar Nginx
  apt:
    pkg: nginx
    state: present
    update_cache: true
  notify:
    - Arranca Nginx
- name: Configuración de H5BP
  copy:
    src: h5bp
    dest: /etc/nginx
    owner: root
    group: root

- name: Quitar la configuración del sitio por defecto
  file:
    dest: /etc/nginx/sites-enabled/default
    state: absent

# El valor de dest va entre comillas al usar una variable dentro
- name: Crear la configuración del sitio
  register: addconfig
  template:
    src: localhost.conf.j2
    dest: '/etc/nginx/sites-available/{{ domain }}.conf'
    owner: root
    group: root
```

```yaml
## El valor de dest y src va entre comillas (variable)
- name: Habilitar la configuración del sitio
  file:
    src: '/etc/nginx/sites-available/{{ domain }}.conf'
    dest: '/etc/nginx/sites-enabled/{{ domain }}.conf'
    state: link

## El valor de dest va entre comillas al usar una variable
- name: Crear la Web root
  file:
    dest: '/var/www/{{ domain }}/public'
    mode: u=rx,g=rx,o=rx
    state: directory
    owner: www-data
    group: www-data
  notify:
    - Recarga Nginx

## El valor de dest va entre comillas al usar una variable
- name: Permisos de la Web Root
  file:
    dest: '/var/www/{{ domain }}'
    mode: u=rx,g=rx,o=rx
    state: directory
    owner: www-data
    group: www-data
    recurse: yes
  notify:
    - Recarga Nginx
```

roleTasks.yml

Roles

José Manuel
Redondo López

- Esta lista de tareas configura Nginx de manera más completa, haciendo lo siguientes pasos
  - **Añadir el repositorio** `nginx/stable`
  - **Instalar e iniciar** Nginx
  - **Añadir archivos** de configuración sacados de H5BP
  - **Deshabilitar la configuración predeterminada** de Nginx
    - Quitando el enlace simbólico al archivo default del directorio `sites-enabled`
  - **Copiar la plantilla de host virtual** `serversample.conf.j2` en la configuración de Nginx
    - Rellanando el valor de las variables necesarias
  - **Habilitar la configuración** del servidor Nginx vinculándola al directorio `sites-enabled`
  - Crear el **directorio raíz web**
  - **Cambiar permisos** (recursivamente) para el directorio raíz del proyecto

ROLES: MÓDULOS COPY, TEMPLATE Y FILE

José Manuel
Redondo López

- Este rol también muestra algunos módulos nuevos y ya mencionados
  - **Copy**: https://docs.ansible.com/ansible/latest/modules/copy_module.html
  - **Template**: https://docs.ansible.com/ansible/latest/modules/template_module.html
  - **File**: https://docs.ansible.com/ansible/latest/modules/list_of_files_modules.html
- Adicionalmente, muestra algunas operaciones interesantes realizadas al crear argumentos para cada módulo
  - **Eliminar archivos**: Asegurarnos de que los archivos están "ausentes" (eliminándolos si existen) a través de `state: absent`
  - Crear un archivo como **enlace simbólico** a través de `state: link`
- En los enlaces de la documentación de cada módulo aparecen más opciones a realizar con los mismos

Ejecución del rol

José Manuel
Redondo López

- Para ejecutar uno o más roles en un servidor se puede usar un **Playbook**
- Dicho **Playbook** debe estar en el mismo directorio que el directorio `roles`
  - Es como el anterior, pero haciendo mención al rol
  - Todo el resto de las cosas que habíamos definido ahora se encuentran en la carpeta adecuada del rol
  - Le llamaremos `server.yml`

```yaml
---
- hosts: nginx_web
  become: yes
  become_user: root
  become_method: sudo
  roles:
    - nginx
```

`server.yml`

```
drwxr-xr-x  4 redondo redondo 4096 Jul 14 15:39 .
drwxr-xr-x 20 redondo redondo 4096 Jul 14 15:33 ..
drwxr-xr-x 10 redondo redondo 4096 Jul 14 15:22 nginx
-rw-r--r--  1 redondo redondo  101 Jul 14 15:39 server.yml
drwxr-xr-x  7 redondo redondo 4096 Jul 14 15:35 ssl-certificate
```

EJECUCIÓN DEL ROL

José Manuel
Redondo López

- `ansible-playbook server.yml`

```
TASK [nginx : Configuración H5BP] ******************************************
ok: [192.168.2.200]

TASK [nginx : Quitar la configuración del sitio por defecto] ***************
ok: [192.168.2.200]

TASK [nginx : Add SFH Site Config] *****************************************
changed: [192.168.2.200]

TASK [nginx : Habilitar la configuración del sitio] ************************
ok: [192.168.2.200]

TASK [nginx : Crear la Web root] *******************************************
ok: [192.168.2.200]

TASK [nginx : Permisos de la Web Root] *************************************
ok: [192.168.2.200]

PLAY RECAP *****************************************************************
192.168.2.200              : ok=16   changed=1    unreachable=0    failed=0    s
kipped=4    rescued=0    ignored=0

operario@server1804:~/ansible-example$
```

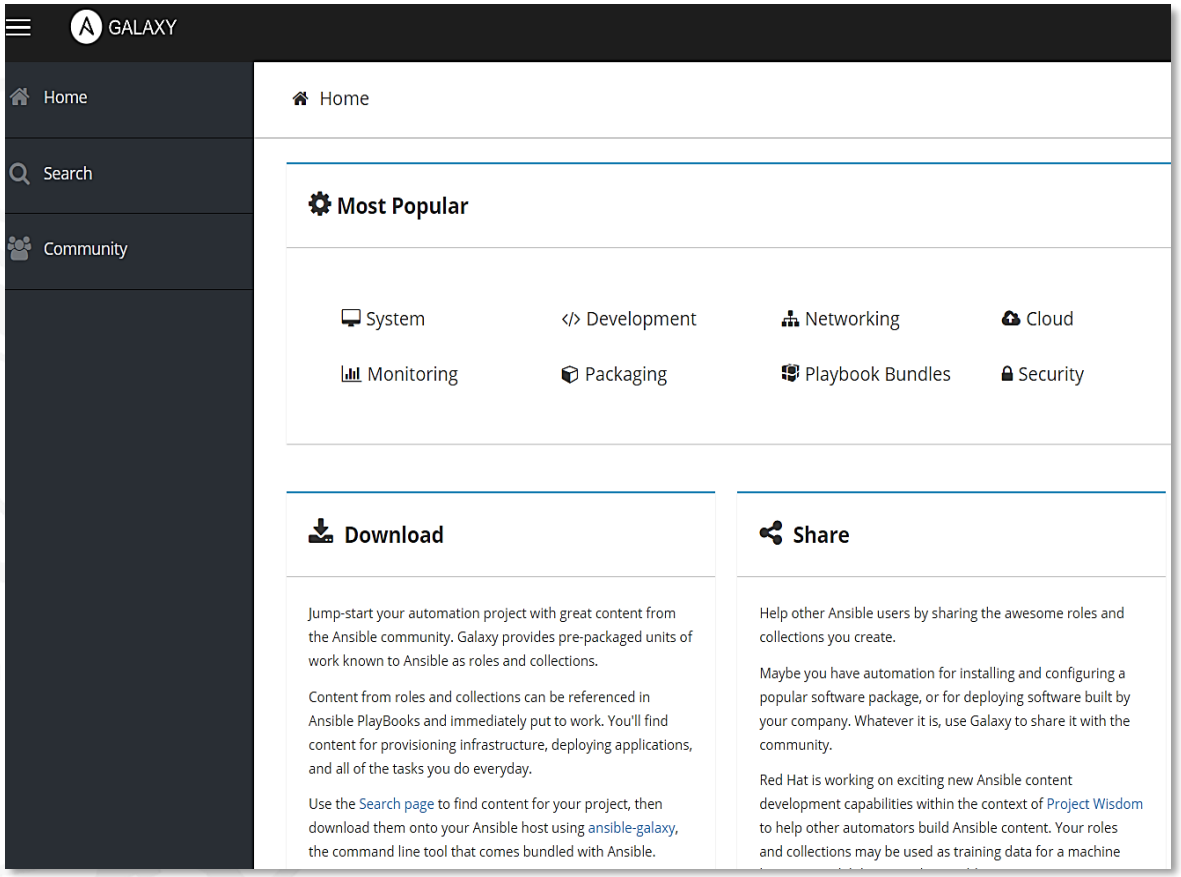
Máquina destino controlada por Ansible

```
redondo@miw:/etc/nginx/sites-enabled$ cat serversample.local.conf
# Fichero-plantilla de configuración templates/localhost.conf.j2
server {
    # Enforce the use of HTTPS
    listen 80 default_server;
    server_name serversample.local;
    return 301 https://$server_name$request_uri;
}
server {
  listen 443 ssl default_server;

  root /var/www/serversample.local/public;
  index index.html index.htm index.php;

  access_log /var/log/nginx/serversample.local.log;
  error_log  /var/log/nginx/serversample.local-error.log error;

  server_name serversample.local;
    charset utf-8;
    include h5bp/basic.conf;
#Estas dos variables están definidas en el rol ssl-certificate #que nos descargamos de GitHub antes
    ssl_certificate        /etc/ssl/localcerts/server.crt;
    ssl_certificate_key    /etc/ssl/localcerts/server.key;
    include h5bp/ssl/ssl_engine.conf;

  location = /favicon.ico { log_not_found off; access_log off; }
  location = /robots.txt  { log_not_found off; access_log off; }
}
```

Ansible Galaxy

José Manuel
Redondo López

- **Repositorio de roles de la comunidad**
  - https://galaxy.ansible.com/
- **Podemos reutilizarlos (todo o partes) en nuestros roles / Playbooks**
  - Son una forma ideal de aprender y de **no tener que construir de cero nuestras operaciones**
    - Alguien pudo haber hecho y necesitado lo mismo que tú
  - Tiene búsqueda y pueden descargarse localmente
  - Gracias al comando `ansible-galaxy role install <nombre del rol>`
- **Al ser contenido de 3ºs, conviene revisarlo a conciencia para ver qué acciones realiza**
  - Y estar seguro de que no rompe ningún servicio instalado o que hagan algo malicioso ☹
- **Tutorial de uso**
  - https://www.scaleway.com/en/docs/how-to-configure-ansible-galaxy/



Al estar dividido en categorías, es más fácil encontrar lo que necesites

Colecciones (Collections)

José Manuel
Redondo López

- **Las colecciones son formas de distribuir contenidos en Ansible**
  - Pueden contener todos los elementos ya vistos: playbooks, roles, módulos, plugins…
- **Se pueden instalar y usar colecciones de Ansible Galaxy con**
  - `ansible-galaxy collection install <espacio de nombres>.<colección>`
  - Con ello **instalaremos todo el contenido de dicha colección**
  - Que se podrá actualizar si hay nuevas versiones añadiendo `--upgrade` al final del comando anterior
- **Los módulos de Ansible más importantes (`ansible.builtin`) y su documentación se han movido a colecciones**
- **Para saber más**
  - En este enlace se explica cómo usar colecciones
    - https://docs.ansible.com/ansible/latest/user_guide/collections_using.html
  - En este se listan todas las colecciones disponibles en la web oficial de Ansible
    - https://docs.ansible.com/ansible/latest/collections/index.html#list-of-collections

¿Crees que lo has entendido todo? ¡Autoevalúate!

José Manuel
Redondo López

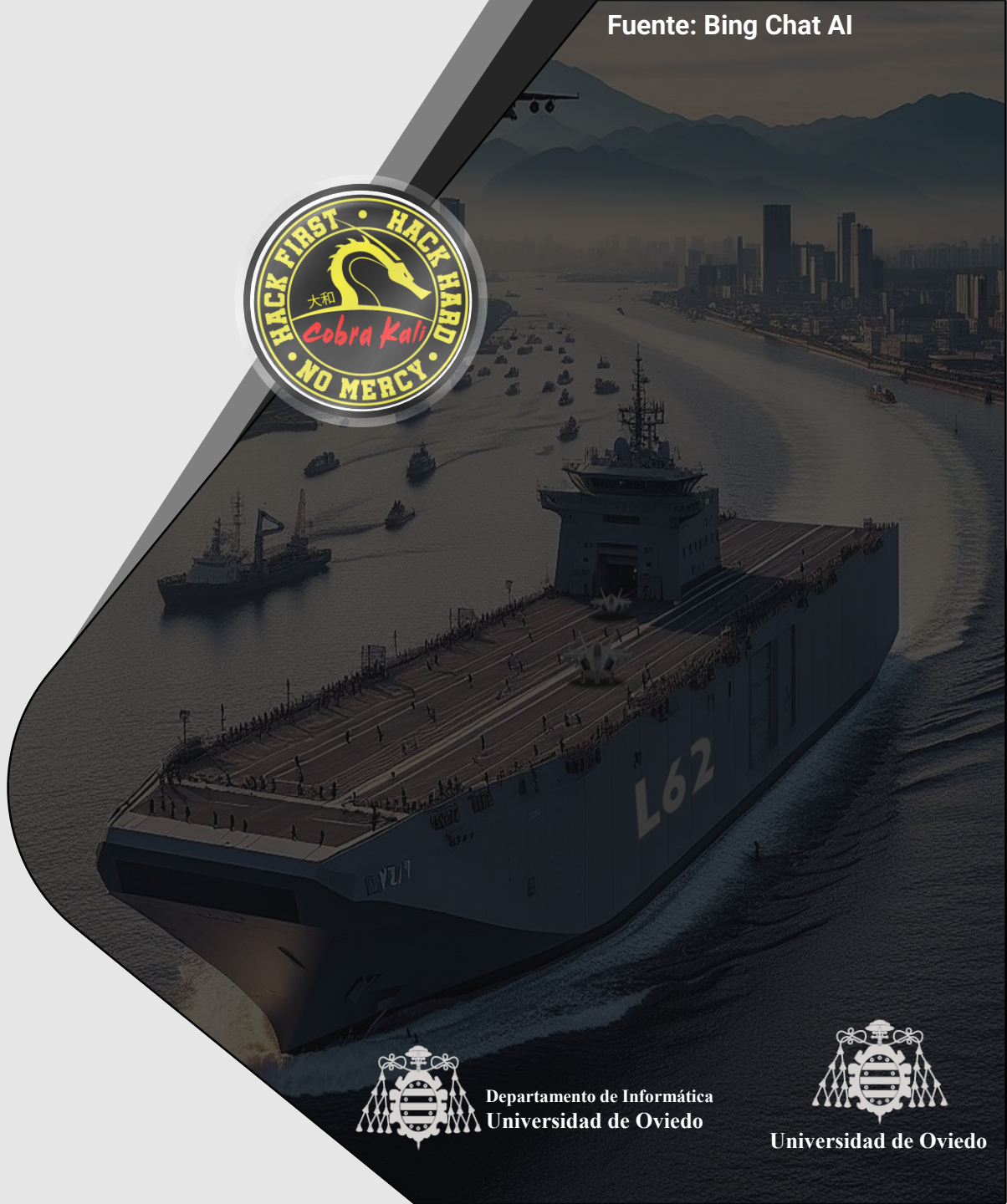
?

- *¿Has entendido cuándo cobran importancia los roles en el trabajo con Ansible?*
- *¿Comprendes que un rol tiene una estructura fija, y que cada una de sus partes tiene un propósito concreto preestablecido?*
- *¿Has entendido cómo crear un módulo desde cero?*
- *¿Sabes qué poner en la carpeta files de un rol?*
- *¿Sabes qué poner en la carpeta handlers de un rol?*
- *¿Entiendes qué son los metadatos de un rol y cómo funcionan sus dependencias?*
- *¿Sabes qué poner en la carpeta vars y en la defaults de un rol?*
- *¿Entiendes cómo funciona una template de un rol de Ansible y para qué sirve?*
- *¿Has entendido qué uso tienen los módulos copy, template y file?*
- *¿Comprendes cómo ejecutar un rol?*
- *¿Tienes clara la función de Ansible Galaxy y como se organiza en colecciones?*

< Ir al Índice

Ansible Vault

Como guardar datos confidenciales en una máquina



Fuente: Bing Chat AI

Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

¿QUÉ VAMOS A VER EN ESTE APARTADO?

José Manuel
Redondo López

- **En este apartado vamos a ver la herramienta necesaria para manejar información secreta como parte de nuestras operaciones de administración con Ansible: Ansible Vault**
  - La **necesidad** de una herramienta como esta
  - **Dónde podemos equivocarnos** con la gestión de secretos
  - **Cómo usar esta herramienta** para hacer las cosas correctamente al escribir nuestro código Ansible

ANSIBLE VAULT

José Manuel
Redondo López

- A la hora de crear plantillas, variables, etc. es frecuente que necesitemos almacenar datos **confidenciales**
- El único dato confidencial que necesitamos hasta ahora a veces era la clave para el `sudo`, y no la guardábamos, pidiéndola al usuario por consola
  - Pero no siempre podremos hacer eso, que además impediría la total automatización del sistema
- **Ansible** tiene una solución para estos casos llamada **Ansible Vault**
  - **Vault** permite cifrar cualquier archivo YAML, como por ejemplo **archivos de variables**
  - No cifra archivos y plantillas, **solo archivos YAML**
- Al crear un archivo cifrado, se pedirá una contraseña para editarlo o al llamar a los **Roles** o **Playbooks** que contienen sus variables cifradas
  - Por tanto, esa contraseña de descifrado debe protegerse a toda costa
- Más información: https://docs.ansible.com/ansible/latest/user_guide/playbooks_vault.html

Ansible Vault

José Manuel
Redondo López

- **La sintaxis de uso es**
  - `ansible-vault [create|decrypt|edit|encrypt|rekey] <ruta del fichero YAML>`
  - `create`: Crear un **nuevo** archivo y cifrarlo
  - `decrypt`: Crear un archivo de texto a partir de un archivo cifrado
  - `edit`: Editar un archivo cifrado ya existente
  - `encrypt`: Cifrar un archivo de texto **existente**
  - `rekey`: Establecer una nueva contraseña en un archivo cifrado
- **Los archivos del Vault suelen contener datos privados que no pueden aparecer en texto plano por los riesgos que supone que personas no autorizadas los lean**
  - Como tokens de acceso a APIs, claves privadas SSH...
  - ¡Su cifrado es extremadamente importante!
  - ¿Qué pasa si subes tu código sin los secretos cifrados a Ansible Galaxy? ¿O al repositorio GitHub de tu aplicación?

Ansible Vault

José Manuel
Redondo López

- **Se puede crear un archivo de variables nuevo en un rol así**
  - `ansible-vault create vars/main.yml`
- **En ese momento se pedirá una contraseña de cifrado para el Vault**
  - Después de introducirla, el archivo se abrirá en el editor predeterminado
- **El editor utilizado se define mediante la variable de entorno `EDITOR`**
  - `EDITOR=nano`
  - `ansible-vault edit vars/main.yml`
  - Otra opción para cambiar el editor es la configuración de perfil/bash de los usuarios
    - Normalmente se encuentra en el `.profile`, `.bashrc` o similar
    - Dentro de la carpeta `$HOME` del usuario concreto
- **En este fichero podemos ahora introducir información que se quedará cifrada**
  - Si son passwords, probablemente tengamos que introducir su hash para poder escribirlas donde corresponda (Ej.: fichero `/etc/shadow`)
  - Un sistema/aplicación segura **NUNCA guarda las claves de sus usuarios en plano**

Ansible Vault: Ejemplo

José Manuel
Redondo López

- Gracias a esto, ahora podremos guardar en el **vault** variables cuyo contenido está cifrado con la clave especificada
  - Por ejemplo, `admin_password` y `deploy_password`
- En el **vault** podemos usarlas en las **Tasks** como variables normales
- Por ejemplo, el texto de la derecha es un fichero `/ansible-example/roles/users/tasks/main.yml` de un nuevo rol `users` que hemos creado

```yaml
---
- name: Crear usuario admin
  user:
    name: admin
    password: '{{ admin_password }}'
    groups: sudo
    append: yes
    shell: /bin/bash
- name: Crear usuario deploy
  user:
    name: deploy
    password: '{{ deploy_password }}'
    groups: www-data
    append: yes
    shell: /bin/bash
```

taskUsers.yml

Ansible Vault: Ejemplo

José Manuel
Redondo López

- **Este nuevo rol usa el módulo de Ansible `user` para crear usuarios**
  - Pasando las **hash** de las contraseñas del archivo cifrado
  - https://docs.ansible.com/ansible/latest/modules/user_module.html
  - También se podría usar el **módulo** `authorized_key`
  - Para agregar una clave publica SSH autorizada en el servidor para cada usuario
    - https://docs.ansible.com/ansible/latest/modules/authorized_key_module.html
- **Editamos el Playbook anterior (`server.yml`) para usar el nuevo rol**
  - No es una dependencia, se ejecutan secuencialmente
  - Tendremos que decirle a Ansible que además ahora **pida la contraseña del Vault** para poder descifrar las variables
  - `ansible-playbook --ask-vault-pass server.yml`

```
---
- hosts: nginx_web
  become: yes
  become_user: root
  become_method: sudo
    roles:
      - nginx
      - users
```

```
admin@server1804:/etc/nginx/sites-enabled$ su admin
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

admin@server1804:/etc/nginx/sites-enabled$ su deploy
Password:
deploy@server1804:/etc/nginx/sites-enabled$ _
```

¿Crees que lo has entendido todo? ¡Autoevalúate!

José Manuel
Redondo López

?

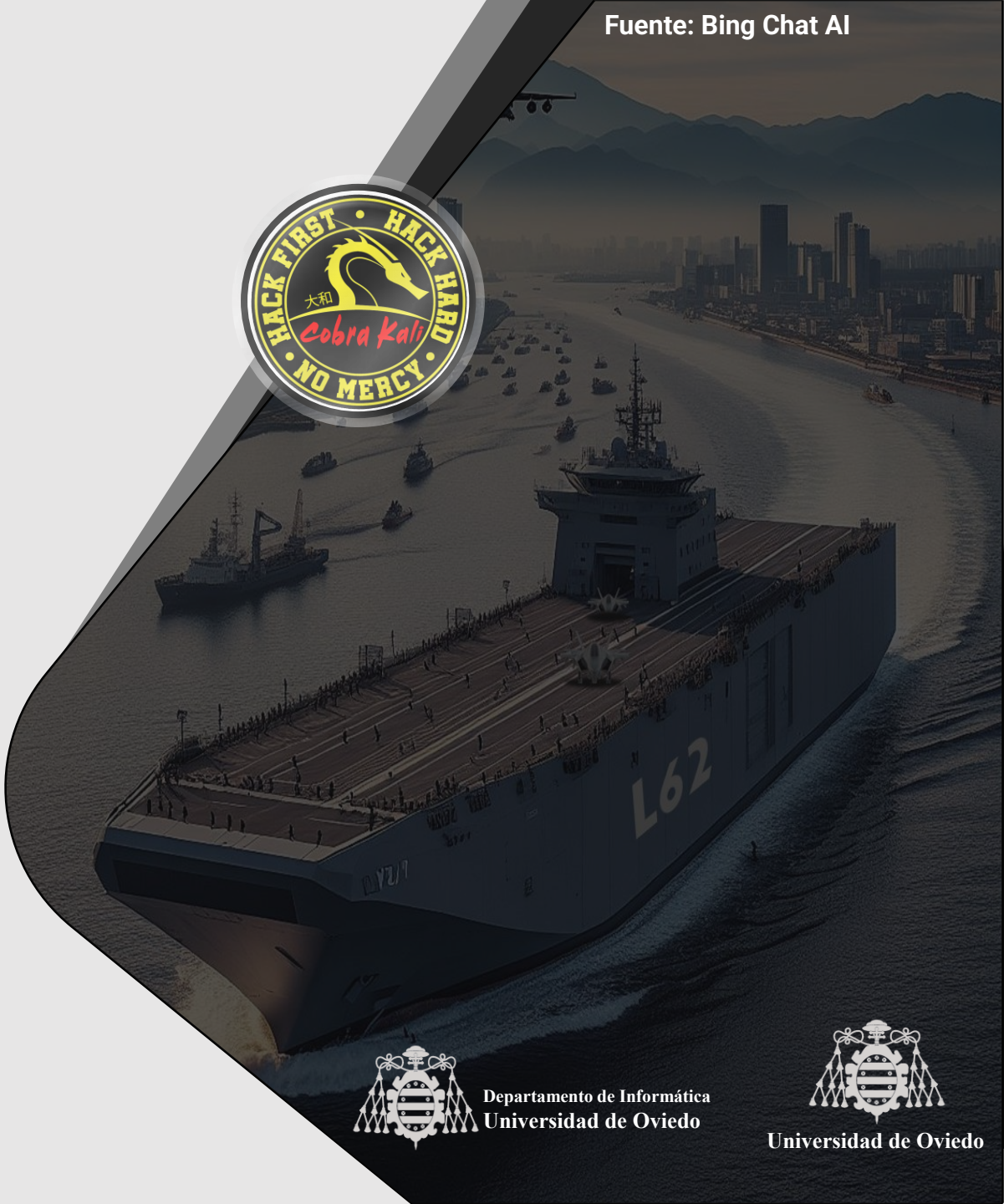
- *¿Entiendes la importancia de una gestión correcta de secretos, y los riesgos que tiene no hacerlo?*
- *¿Has entendido las operaciones básicas de creación, cifrado, descifrado y gestión de un **vault**?*
- *¿Tienes claro cómo editar un fichero de variables cifradas?*
- *¿Entiendes cómo lanzar un rol o **playbook** que use variables cifradas?*
- *¿Comprendes cómo funciona el módulo **user** de **Ansible**?*

Recapitulando...

José Manuel
Redondo López

- **Por tanto, hasta ahora hemos**
  - Instalado Ansible
  - Configurado un inventory de Ansible simple
  - Ejecutado comandos idempotentes en varios servidores
  - Creado un Playbook básico para ejecutar varias tareas, usando Handlers
  - Abstraído los distintos conceptos en Roles para mejorar la organización
    - Incluyendo cómo establecer dependencias
    - Registrar esas dependencias de las tareas y ejecutar otras tareas solo si son correctas
    - Usar plantillas, archivos y variables en nuestras tareas
  - Uso y extracción de Facts
  - Uso de Ansible Vault para usar variables de forma segura
- **Con esto cubrimos todo lo básico de Ansible**
  - Aunque la herramienta tiene muchas más opciones
  - En los anexos hay dónde buscar más información

< Ir al Índice



Ansible, Vagrant y Docker

Usando conjuntamente ambas tecnologías

¿QUÉ VAMOS A VER EN ESTE APARTADO?

José Manuel
Redondo López

- **Dado que Ansible es una herramienta de administración remota de infraestructuras, vamos a ver cómo integrarla con las herramientas para crear infraestructura que hemos visto hasta ahora**
  - Integrar Ansible con Vagrant **para que haga de provisionador** del mismo
  - **Controlar la instalación y ejecución** de contenedores Docker con Ansible
  - Entender que, al final, a Ansible le da igual que manejes máquinas virtuales, contenedores u otra cosa: **solo necesita una IP y acceso SSH**
    - Por eso **es más correcto usar el término endpoint** para referirse a lo que Ansible puede manejar
    - Así lo haremos en las prácticas de este tema

Interacción Ansible-Vagrant

José Manuel
Redondo López

- **Es posible provisionar MVs Vagrant con Ansible**
  - Disponemos así de toda su funcionalidad para hacerlo, siendo una combinación muy potente
- **Lanzar el provisioner es muy sencillo,**
  - Indicando el fichero con el playbook del que leer la configuración
- **Más opciones del provisioner**
  - https://www.vagrantup.com/docs/provisioning/ansible
- **Ejemplos de estas combinaciones**
  - https://github.com/geerlingguy/ansible-vagrant-examples

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
Vagrant.configure("2") do |config|
  config.vm.box = "generic/ubuntu1804"

  config.vm.provider :virtualbox do |v|
    v.memory = 1024
  end

  # provisioner Ansible
  config.vm.provision :ansible do |ansible|
    ansible.playbook = "playbook.yml"
    ansible.become = true
  end
end
```

Interacción Ansible - Docker

José Manuel
Redondo López

- **Ansible** también permite gestionar la configuración de contenedores **Docker**
- Existe un módulo `docker` que integra sus funcionalidades con las de la plataforma de contenedores que ya hemos visto
  - https://docs.ansible.com/ansible/latest/index.html
- Pero para poder usarlo necesitaremos
  - **Docker** instalado en las máquinas a gestionar
  - **Ansible** en la máquina que vaya a lanzar los **scripts**
    - Que ya lo tenemos de nuestras pruebas anteriores

Instalando Docker con Ansible

José Manuel
Redondo López

- **Docker es un prerrequisito**
  - Podemos usar un **Playbook** que lo instale
- **Sería una tarea previa al manejo de Docker y Ansible**
- **Nos permite ilustrar algunos de los conceptos vistos antes**
  - Creamos un **Playbook** `dockersample.yml` con el siguiente contenido
  - Lo haremos dentro de una carpeta llamada `docker_ansible`
  - También podríamos hacerlo con un rol
    - Ej.: https://github.com/marvinpinto/ansible-role-docker-nginx

```yaml
- hosts: nginx_web
  become: yes
  become_user: root
  become_method: sudo
  tasks:
  - name: Clave GPG de Docker
    apt_key: url=https://download.docker.com/linux/ubuntu/gpg
  - name: Instala prerrequisitos
    apt:
      name: ['apt-transport-https','ca-certificates','curl',
      'gnupg2','software-properties-common']
      state: present
      update_cache: yes
  - name: Repositorio de Docker
    apt_repository:
      repo: deb [arch=amd64]
      https://download.docker.com/linux/{{ansible_distribution|lower}}
      {{ansible_distribution_release}} stable
  - name: Instala Docker
    apt:
      name: ['docker-ce','docker-ce-cli','containerd.io']
      state: present
    register: dockerinstalled
```

dockersample.yml

Instalando Docker con Ansible

José Manuel
Redondo López

- Añadimos ahora al **playbook** la instalación de una librería `docker-py` especial
  - En los host que tienen **Docker**
  - Es necesaria para coordinar **Ansible** y **Docker**
- Para ello es necesario `pip`
  - Por lo que también tenemos que instalarlo antes que la librería anterior
  - Con ello tendremos acceso a este módulo
    - https://docs.ansible.com/ansible/latest/modules/pip_module.html
- Por tanto, añadirnos al fichero anterior dos tareas más
- La librería debe instalarse siempre después que el propio **Docker**

```
… (resto del fichero dockersample.yml)
  - name: Instalar pip
    apt:
      name: python3-pip
      state: present
  - name: Instalar paquete docker-py
    pip:
      name: docker
    when: dockerinstalled is success
```

Playbook para contenedores del Docker Hub

José Manuel
Redondo López

- **Creamos un contenedor con un Nginx utilizable externamente**
  - Con la imagen oficial `nginx` de Docker Hub
  - **Además del módulo** `docker_container`
  - https://docs.ansible.com/ansible/latest/modules/docker_container_module.html
- **Para ello indicamos**
  - El **nombre de la imagen** del Docker Hub
    - Para usar la última versión de la imagen se añade el prefijo `:latest` a su nombre
  - Que obtenga (`pull`) siempre su última versión
  - Un **nombre** del contenedor (obligatorio)
  - Los **puertos que expone** el contenedor (80)
  - El **mapeo de puertos** del host a puertos expuestos por el contenedor (`80:80`)
  - Un parámetro de estado que le diga a Ansible que queremos la imagen en ejecución (`started`)

```
… (resto del fichero dockersample.yml)
 - name: Ejecutando el contenedor con una imagen del Docker Hub
     docker_container:
       image: nginx:latest
       pull: true
       name: docker_nginx
       exposed_ports:
         - '80'
       published_ports:
         - '80:80'
       state: started
```

COMPROBANDO QUE LA EJECUCIÓN ES CORRECTA

José Manuel
Redondo López

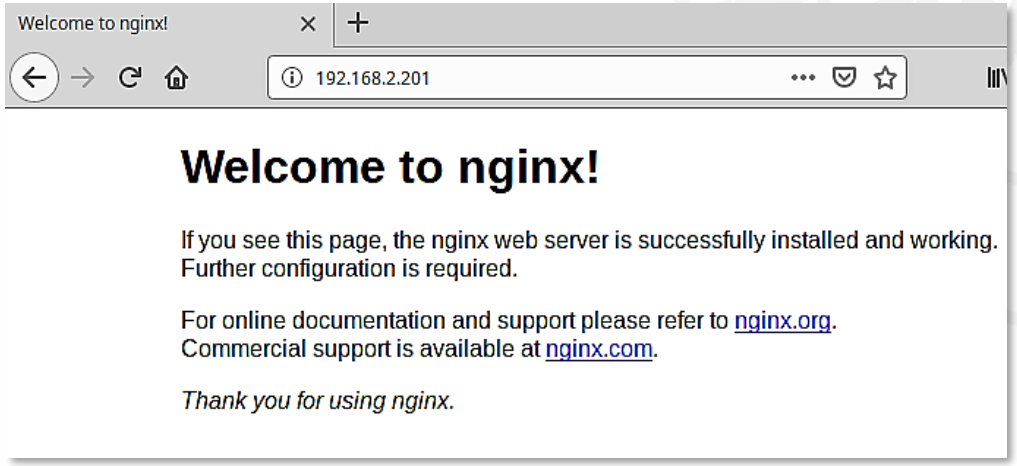
- **Podemos comprobar si funciona de 2 formas**
  - Viendo si la máquina sirve ahora `nginx`, (solo podría desde el contenedor)
  - Obteniendo información de los contenedores arrancados
    - **Eso puede hacerse con el módulo `docker_container_info`**
    - https://docs.ansible.com/ansible/latest/modules/docker_container_info_module.html#return-values
    - Usamos su resultado para hacer un par de tareas que nos impriman información del contenedor a través de objetos `debug.msg`
    - Y así vemos cómo podemos imprimir información dentro de **Playbooks**

```
… (resto del fichero dockersample.yml)
 - name: Comprobar que el contenedor funciona
    docker_container_info:
      name: docker_nginx
    register: info
  - name: Imprime si el contenedor existe
    debug:
      msg: "El contenedor {{ 'existe' if
info.exists else 'no se ha creado' }}"
  - name: Imprime el estado del contenedor
    debug:
      msg: "El estado del contenedor es {{
info.container['State']['Status'] }}"
    when: info.exists
```

Ejecución del Playbook

José Manuel
Redondo López

- Finalmente, solo nos queda la ejecución del **Playbook** de la misma forma que ya vimos
  - `ansible-playbook dockersample.yml`
- Para comprobar que funciona
  - `curl <IP del servidor>`
- O bien usar un browser




```
operario@server1804:~/docker_ansible$ ansible-playbook dockersample.yml --ask-become-pass
BECOME password:

PLAY [nginx_web] ***************************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.2.201]

TASK [Clave GPG de Docker] *****************************************************
changed: [192.168.2.201]

TASK [Instala prerrequisitos] **************************************************
 [WARNING]: Could not find aptitude. Using apt-get instead

changed: [192.168.2.201]

TASK [Repositorio de Docker] ***************************************************
changed: [192.168.2.201]

TASK [Instala Docker] **********************************************************
changed: [192.168.2.201]

TASK [Instalar pip] ************************************************************
changed: [192.168.2.201]

TASK [Instalar paquete docker para Python] *************************************
changed: [192.168.2.201]

TASK [Ejecutando el contenedor sacando la imagen del Docker Hub] ***************
changed: [192.168.2.201]

TASK [Comprobar que el contenedor funciona] ************************************
ok: [192.168.2.201]

TASK [Imprime si el contenedor existe] *****************************************
ok: [192.168.2.201] => {
    "msg": "El contenedor existe"
}

TASK [Imprime el estado del contenedor] ****************************************
ok: [192.168.2.201] => {
    "msg": "El estado del contenedor es running"
}
```

¿Y SI QUIERO USAR MIS PROPIAS IMÁGENES?

José Manuel
Redondo López

- Si no queremos usar una imagen del **Docker Hub** y queremos arrancar una que hayamos configurado, podemos usar el **módulo** `docker_image`
  - https://docs.ansible.com/ansible/latest/modules/docker_image_module.html
- Para ilustrar este proceso haremos un nuevo fichero llamado `dockersample2.yml`
  - Basado en el que hicimos antes
- La mayor parte de acciones ya estarán hechas
  - Instalar Docker, scripts de Python…
  - ¡Pero no importa gracias a la idempotencia de Ansible!
  - **Recuerda: ¡Todo lo que ya esté hecho no se volverá a hacer!**
- Usaremos este proceso para ver cómo trabajar con una máquina destinada a ser interactiva
- Todo este Playbook puede ser convertido fácilmente en un rol

¿Y SI QUIERO USAR MIS PROPIAS IMÁGENES?

José Manuel
Redondo López

- **Una Dockerfile construirá la imagen en el destino**
- **Pero el destino debe tener Docker instalado**
  - Podemos crear el fichero en destino
  - O bien (**más práctico**) crearlo en la máquina que ejecuta Ansible y copiarlo a la ruta deseada mediante el módulo de Ansible `copy`
    - https://docs.ansible.com/ansible/latest/modules/copy_module.html
    - En nuestro caso la copiaremos en la carpeta `docker` dentro de la cuenta de usuario que usamos para acceder vía SSH

Dockerfile

```
FROM ubuntu
RUN apt-get update
RUN apt-get install -y nmap
```

dockersample2.yml

```
… (MISMO CONTENIDO QUE DOCKERSAMPLE.YML HASTA LA LÍNEA "WHEN: DOCKERINSTALLED IS SUCCESS")
  - NAME: COPIAR LA DOCKERFILE AL DESTINO
    COPY:
      SRC: DOCKERFILE
      DEST: DOCKER/ #SI ACABA EN / LO CREA
      MODE: U+RW
```

¿Y SI QUIERO USAR MIS PROPIAS IMÁGENES?

José Manuel
Redondo López

- **El siguiente paso es construir la imagen con** `docker_image`
  - Y arrancar un contenedor de la misma
- **Para construirla tenemos que dar**
  - Un nombre a la imagen (`ubuntu_nmap_instance`)
  - Una acción en `source` (en nuestro caso `build`)
  - Dónde está la **Dockerfile** de la imagen
    - La tarea anterior la copió a la carpeta remota `docker`
- **Para ejecutar un contenedor tenemos que modificar el ejemplo de antes**
  - La imagen **no** se saca del **Docker Hub** (`pull:no`)
  - La imagen **no es un servicio**, sino que está pensada para interactuar con ella
    - `interactive: yes`
    - `tty: yes`
- **El resto es igual**

```
  - name: Construir la imagen con la Dockerfile
    docker_image:
      name: ubuntu_nmap
      source: build
      build:
        path: docker
      state: present
    register: imagebuilt
  - name: Ejecutando un contenedor de la imagen creada
    docker_container:
      image: ubuntu_nmap
      name: ubuntu_nmap_instance
      pull: no
      state: started
      interactive: yes
      tty: yes
```

Ejecución del nuevo Playbook

José Manuel
Redondo López

- La ejecución de este Playbook es exactamente igual a la anterior
- Hemos dejado los mecanismos de comprobación del contenedor anteriores
  - Para que la ejecución muestre si ha tenido éxito

```
TASK [Instalar paquete docker para Python] *************************************************************
ok: [192.168.2.201]

TASK [Copiar la Dockerfile al sistema remoto] **********************************************************
ok: [192.168.2.201]

TASK [Construir la imagen con la Dockerfile] ***********************************************************
 [WARNING]: The default for build.pull is currently 'yes', but will be changed to 'no' in Ansible 2.12. Please
set build.pull explicitly to the value you need.

ok: [192.168.2.201]

TASK [Ejecutando un contenedor de la imagen creada] ****************************************************
changed: [192.168.2.201]

TASK [Comprobar que el contenedor funciona] ************************************************************
ok: [192.168.2.201]

TASK [Imprime si el contenedor existe] *****************************************************************
ok: [192.168.2.201] => {
    "msg": "El contenedor existe"
}

TASK [Imprime el estado del contenedor] ****************************************************************
ok: [192.168.2.201] => {
    "msg": "El estado del contenedor es running"
}

PLAY RECAP *********************************************************************************************
192.168.2.201              : ok=13   changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignore
d=0
```

Ejecución del nuevo Playbook

José Manuel
Redondo López

- Si accedemos a la máquina destino podemos ver que el contenedor está en marcha
- Y también podemos acceder a una sesión interactiva en el mismo
  - `sudo docker exec –it ubuntu_nmap_instance /bin/bash`
- Y comprobar que el programa instalado (`nmap`) está disponible
- Con esto, tenemos el contenedor desplegado y funcionando

```
operario@server1804:~$ sudo docker exec -it ubuntu_nmap_instance /bin/bash
Nmap 7.60 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only scan specified ports
    Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
  --exclude-ports <port ranges>: Exclude the specified ports from scanning
  -F: Fast mode - Scan fewer ports than the default scan
  -r: Scan ports consecutively - don't randomize
  --top-ports <number>: Scan <number> most common ports
--More--
```

Controlar sistemas remotos con Ansible

José Manuel
Redondo López

- **Hemos visto como construir imágenes y contenedores con Ansible usando módulos concretos**
- **¿Y qué pasa si necesitamos aprovisionamiento?**
  - ¡Qué podemos hacerlo de igual forma que cualquier otro sistema!
- **Necesitamos**
  - Acceso vía SSH al destino con un usuario (seguramente que tenga privilegios de `root`)
  - Automático (certificados) o manual (preguntando password)
  - Saber la IP de las máquinas/contenedores LXD/contenedores Docker destino
  - ¡Solo con eso, ya podemos provisionar lo que sea!
- **Esa es la "magia" de Ansible**
  - Controlar endpoints de cualquier tipo sin necesidad de instalarles ningún paquete "cliente"
  - Gracias a eso, es más flexible que otras opciones que sirven para lo mismo
  - Y, por tanto, **podría usarse para administrar contenedores LXD sin cambios** (solo necesitamos una IP del contenedor y acceso SSH con privilegio suficiente al mismo)

¿Crees que lo has entendido todo? ¡Autoevalúate!

José Manuel
Redondo López

?

Ansible - Vagrant

- *¿Entiendes qué es posible ejecutar un provisioner Ansible desde Vagrant y así unir los contenidos vistos en ambos temas?*
- *¿Comprendes cómo eso te da acceso a todas las características de automatización de ambos y que así puedes desplegar infraestructuras de MVs automatizadas mucho más potentes?*

Ansible - Docker

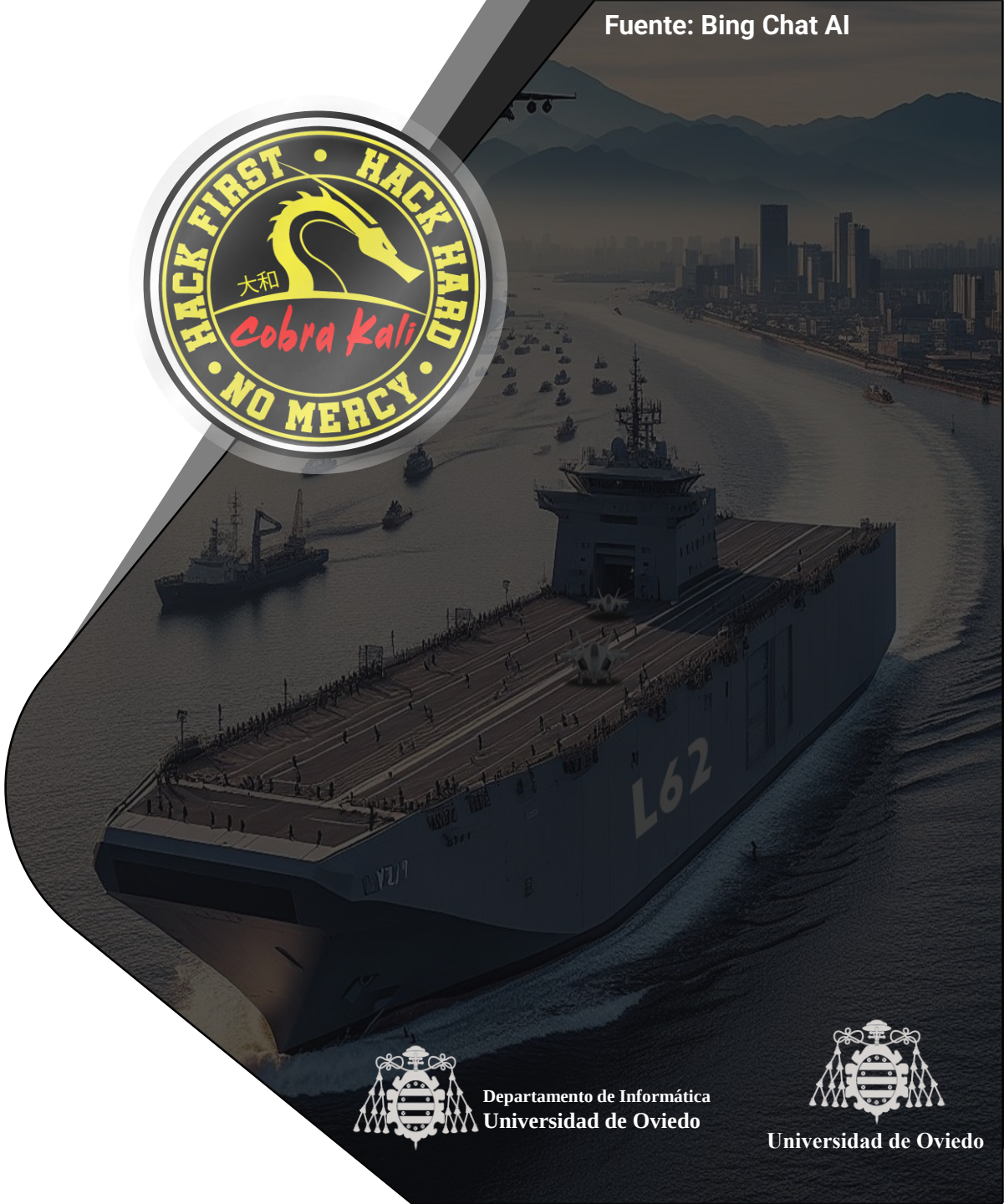
- *¿Entiendes que Ansible tiene módulos especiales para gestionar imágenes y contenedores Docker, así como el sistema de contenedores en sí?*
- *¿Crees que sabrías instalar Docker con Ansible?*
- *¿Crees que sabrías usar una imagen del Docker Hub con Ansible para desplegar un contenedor?*
- *¿Crees que sabrías usar una imagen de Docker tuya propia para desplegar un contenedor?*
- *¿Recuerdas cómo indicar si un contenedor va a ser o no interactivo a Ansible?*
- *¿Has entendido las formas de hacer testing para ver si los contenedores se han desplegado?*
- *¿Comprendes que Ansible te permite abstraerte del endpoint que manejas, y solo entiende de IPs y acceso vía SSH?*

Para saber más...

José Manuel
Redondo López

- **Más información sobre Ansible y Docker**
  - https://www.ansible.com/integrations/containers/docker
  - https://docs.ansible.com/ansible/latest/scenario_guides/guide_docker.html
- **Documentación oficial**
  - https://docs.ansible.com/
- **Tutoriales avanzados**
  - https://serversforhackers.com/c/an-ansible2-tutorial
  - https://likegeeks.com/es/tutorial-de-ansible/
- **Libro**
  - https://book.serversforhackers.com/
- **Repositorio de GitHub con roles utilizables**
  - https://github.com/Servers-for-Hackers

INTRODUCCIÓN A ANSIBLE



Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo