Ansible (I)

Introducción

Ansible es una herramienta de administración remota que permite permite **definir estados** ("que esté instalada esta herramienta", o "que esté configurada aquella interfaz de red", ...) y llevar a los equipos hasta dicho estado realizando las operaciones que sean necesarias. Para ello únicamente debe instalarse en la máquina que controla a las demás, ya que accede a los equipos remotos utilizando simplemente **SSH** (esto implica, eso sí, que Ansible deberá autenticarse en cada uno de los equipos remotos a administrar mediante un determinado usuario existente en ellos y, preferiblemente, ese usuario deberá tener privilegios de "root" vía *sudo* para poder realizar la mayoría de operaciones importantes del sistema).

> NOTA: Por defecto Ansible utiliza concretamente OpenSSH como librería de transporte, aunque puede utilizar otras si así se le indica (como Paramiko, http://www.paramiko.org , por ejemplo)

Simplificando mucho, podríamos decir que Ansible actúa como un cliente SSH masivo capaz de conectarse en paralelo a múltiples servidores SSH (los hosts a administrar) y realizar en cada uno de ellos la operación deseada. Para ello, Ansible ofrece diferentes módulos que permiten realizar diferentes operaciones predefinidas de forma sencilla sin la necesidad de tener que escribir complejos comandos o shells scripts. Ejemplos de módulos son: "apt" (para instalar paquetes .deb en los hosts remotos), "service" (para administrar servicios en los hosts remotos), "file" (para cambiar los atributos de ficheros en los hosts remotos), etc, etc. La mayoría de comandos Unix tienen asociado un módulo Ansible equivalente.

Hay que saber, por otro lado, que tras conectar mediante SSH en un host remoto, Ansible abre en él una subshell (como root si la operación lo requiere) para ejecutar allí dentro un intérprete de **Python** que se encargará de ejecutar un determinado script que representa la implementación real del módulo indicado. Esto quiere decir que, además de tener en marcha un servidor SSH en todas los hosts remotos a administrar, también será necesario tener instalado un intérprete de Python (mediante el paquete llamado "python3")

Por otro lado, un aspecto muy importante de Ansible es que todas las operaciones realizadas en los hosts remotos son **idempotentes**. Esto significa que no hay que pensar en condiciones: se definen las operaciones a realizar y únicamente producirán efecto aquellas que sean necesarias (por no estar el host remoto de acuerdo al estado especificado) dejándose sin tocar los hosts que ya estén en dicho estado.

Un concepto importante en Ansible son los "playbooks". Un "**playbook**" no es más que un fichero de texto donde se especifican (mediante una sintaxis llamada YAML, http://yaml.org) todos los detalles necesarios para que Ansible pueda, en el momento de repasar dicho "playbook", saber qué determinado conjunto de operaciones deberá realizar (mediante el uso de qué módulos determinados) sobre qué determinado conjunto de hosts remotos. Ansible ejecutará el "playbook" secuencialmente y se asegurará de que el estado de cada comando sea el deseado en todos los hosts antes de pasar al siguiente (esto es lo que hace que Ansible sea idempotente: si se cancela la ejecución del "playbook" a la mitad y se reinicia más tarde, solo los comandos que no se hayan completado previamente se ejecutarán

Los "playbooks" permiten crear instrucciones complejas y si no se tiene cuidado pueden volverse difíciles de manejar, lo que nos lleva a otro concepto: "**roles**". Los roles añaden organización a los "playbooks" porque permiten dividir sus instrucciones y bloques en trozos más pequeños reutilizables (muy parecido a una función en términos de programación). Esto hace posible compartir roles en diferentes "playbooks" sin duplicar código.

> NOTA: Ansible no és l'única eina d'administració remota de sistemes. Altres alternatives són:
>
> CFEngine (https://cfengine.com)
> Chef (https://www.chef.io/chef)
> Puppet (https://puppet.com)
> Salt (https://saltstack.com)
> Foreman (https://www.theforeman.org). Aquest inclou a més un servidor PXE integrat

Instalación y configuración general

Una vez instalado Ansible mediante el comando *sudo dnf install ansible-core sshpass* (en Fedora) o *sudo apt install ansible-core sshpass* (en Ubuntu), lo primero que querremos hacer es estudiar su archivo de configuración general, que es **"/etc/ansible/ansible.cfg"**. No obstante, es posible que no exista porque en la mayoría de ocasiones los valores de configuración predefinidos ya son los adecuados. En el caso de necesitar, de todas maneras, modificar algún comportamiento concreto de Ansible, deberemos entonces crear dicho fichero e incluir en él las directivas específicas que queramos personalizar. A continuación se mencionan algunas directivas que tal vez haya que añadir en algún momento:

*Directiva *forks* (dentro de la sección [defaults]) : Indica el número máximo de "víctimas" a las que Ansible conectará a la vez para ejecutar una determinada tarea. Si el número total de "víctimas" es mayor que ese número, cuando una de las "víctimas" elegida inicialmente haya finalizado la ejecución de esa tarea, Ansible pasará a conectarse a otra máquina nueva para ejecutar dicha tarea allí, y así sucesivamente, siempre manteniendo el número de conexiones paralelas como mucho en el número dado por esta directiva. A mayor número, menor tiempo empleado en finalizar todo, obviamente, pero también más estrés para la máquina controladora.

*Directiva *host_key_checking* (dentro de la sección [defaults]) : Todos los servidores SSH tienen una "clave de host" SSH asociada. Esta clave actúa como una firma que identifica de forma única al servidor y sirve para prevenir ataques MitM porque permiten al cliente verificar que el servidor que dice ser el servidor deseado realmente lo sea. Esto significa que el cliente (en este caso, Ansible) debe tener la clave de host antes de intentar conectarse al host. Este paso debe realizarse solo una vez, pero debe realizarse antes de que Ansible se conecte por primera vez al servidor. La forma más sencilla de tener la clave de host en la máquina de Ansible es conectarse con el cliente *ssh* normal y responder "sí" a la pregunta mostrada; otra "solución" sería deshabilitar el requisito de tener las claves de host (¡pero cuidado, esto es peligroso!); esto se puede hacer asignando el valor falso (*False*) a la directiva *host_key_checking* (su valor predeterminado es *True*).

*Directiva *retry_files_enabled* (dentro de la sección [defaults]) : Cuando un "playbook" falla, por defecto se crea un archivo ".retry" (con el mismo nombre que el libro de jugadas en cuestión) dentro de la misma carpeta donde reside ese "playbook". Este archivo ".retry" contiene las direcciones IP de las máquinas "víctimas" infractoras, una por línea. Se puede deshabilitar esta función configurando *retry_files_enabled* a *false*. Por otro lado si vale *true*, se puede cambiar la ubicación del archivo ".retry" configurando la ruta absoluta del nuevo archivo ".retry" como el valor de la opción *retry_files_save_path* (si esta ruta contiene una carpeta inexistente, se creará).

Para hacer la manipulación de este archivo más fácil, existe la posibilidad de utilizar el comando *ansible-config* (un comando especialmente diseñado para gestionar la configuración de Ansible) de tal manera que genere un archivo "/etc/ansible/ansible.cfg" incluyendo todas las directivas existentes (y con su valor actual) pero comentadas. Así, tan solo deberemos buscar la directiva a modificar en cuestión, modificarla y descomentarla. Concretamente, el comando a ejecutar es ***ansible-config init --disabled | sudo tee /etc/ansible/ansible.cfg***

NOTA: És possible tenir una configuració d'Ansible personalitzada que sobreescrigui la configuració general definida a l'arxiu "/etc/ansible/ansible.cfg" mitjançant l'arxiu "**~/.ansible.cfg**", el qual contindrà una configuració particular per l'usuari concret que executi Ansible. D'altra banda, també es pot utilitzar un arxiu "**/ruta/qualsevol/ansible.cfg**" per tal d'establir una configuració particular per (només) el cas en què Ansible s'executi dins de la ruta indicada (/ruta/qualsevol), sobreescrivint en aquest cas tant a "~/.ansible.cfg" com a "/etc/ansible/ansible.cfg". En qualsevol cas, sempre es podrà indicar un arxiu de configuració diferent (que tindrà prioritat sobre tota la resta) indicant la seva ruta completa com a valor de la variable d'entorn **ANSIBLE_CONFIG**.
NOTA: A la pàgina https://docs.ansible.com/ansible/latest/reference_appendices/config.html#common-options podeu trobar la llista de totes les directives de configuració, juntament amb la seva explicació, el seu valor per defecte i la seva ubicació dins de l'arxiu "/etc/ansible/ansible.cfg". Allà també apareixen llistades les possibles variables d'entorn que es podrien fer servir per sobreescriure en un moment puntual la configuració indicada al fitxer.
NOTA: A la pàgina https://docs.ansible.com/ansible/latest/reference_appendices/config.html#environment-variables podeu trobar la llista de totes les variables d'entorn que Ansible reconeix (i aplica el seu valor) a l'hora d'executar-se

Inventario

La lista de hosts reconocidos por Ansible se ha de indicar en el archivo "**/etc/ansible/hosts**". A continuación se muestra un ejemplo de contenido autoexplicativo de este archivo:

[grupo1]
192.168.0.111
192.168.0.123
#Para indicar un rango de IPs
192.168.1.[01:50]
...
#Si la asociación nombre<-> IP ya está definida en el archivo "/etc/hosts" de la máquina donde se #ejecutará Ansible (o es resoluble mediante DNS), en vez de IPs se puede indicar directamente los #nombres de la máquinas víctimas, así:
nombre.Maquina.dom.inio
#También se puede indicar el nombre (inventado) de una máquina y asociarla a una determinada IP (o nombre real) mediante el uso de la variable *ansible_host*, así:
unNombre ansible_host=192.168.1.222

[grupo2]
#Para indicar que el servidor SSH de ese host escucha en otro puerto diferente del puerto por defecto #(que es 22 y está indicado en "ansible.cfg") se puede usar la variable *ansible_port* así:
192.168.0.222 ansible_port=1234
#Para indicar el usuari a usar por defecto contra el servidor SSH (que por defecto es root, y está #indicado en "ansible.cfg") se puede usar la variable *ansible_user* así:
192.168.0.231 ansible_user=pepa
#Para realizar una conexión a la máquina local y, por tanto, no usar SSH (por defecto el valor de la #variable *ansible_connection* es "ssh". Otros valores posibles son "docker", "podman", "lxd", ...)
#Notar que en el caso de querer usar 127.0.0.1 (o localhost) como víctima no es necesario indicar esa #IP en el inventario porque ya se reconoce automáticamente (y Ansible ya le elige conexión "local")
127.0.0.1 ansible_connection=local

#Si queremos indicar el mismo valor para determinadas variables sin tenerlas que escribir host a #host, se pueden asignar a grupos enteros así (también se podría hacer [all:vars], ver nota siguiente)
[grupo2:vars]
ansible_port=2222
ansible_user=pepe

#Si queremos generar grupos que contengan otros grupos, se puede hacer así
[grupoPadre:children]
grupo1
grupo2

NOTA: Existe siempre un grupo predefinido llamado "all" que contiene todos los grupos indicados en el inventario. Así pues, el orden de sobreescritura de una variable, si se escribe la misma en diferentes lugares del inventario es el siguiente: *all → grupo_padre → grupo_hijo → victima_individual*. Ver cuadro de la página siguiente para una jerarquía más completa.
NOTA: Se podría utilizar otro fichero diferente de "/etc/ansible/hosts" si se indica su ruta en la directiva *inventory* del fichero "ansible.cfg" (bajo la sentencia *[defaults]*). O indicando la ruta del fichero deseado como valor de la variable de entorno **ANSIBLE_INVENTORY** También se puede indicar mediante el parámetro *-i* del comando *ansible/ansible-playbook* en el momento de lanzar la/s operación/es
NOTA: Per inventoris complexes, pot ser útil fer servir la comanda ***ansible-inventory {--list [--yaml]|--graph} [-i /ruta/fitxer/inventori]***, la qual mostra d'una manera visual l'estructura jeràrquica de grups existents dins de l'inventori i dels dispositius que en formen part

En el apartado "List of behavioral inventory parameters de la página http://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html aparecen listadas todas las posibles variables "ansible_*" que se pueden indicar en el inventario, ya sea asociándolas a un único host o a un grupo (o varios). Además de las ya indicadas, entre las más interesantes, podemos encontrar:

ansible_ssh_pass : Contrasenya SSH a usar. Si no es vol escriure en texte pla, caldrà usar un "Ansible Vault", que ja veurem. Una altra opció és no indicar aquesta opció i deixar llavors que Ansible la demani interactivament en el moment d'executar-lo

ansible_become : Si val *yes*, es podrà fer servir *sudo/su* a les màquines víctimes corresponents si cal

ansible_become_user : Indica l'usuari en el què es convertirà Ansible en executar *sudo/su*. Per defecte és "root"

ansible_become_pass: Contrasenya *sudo/su* a usar. Si no es vol escriure en texte pla, caldrà usar un "Ansible Vault". Una altra opció és no indicar aquesta opció i deixar llavors que Ansible la demani interactivament en el moment d'executar-lo

ansible_python_interpreter : Indica la ruta absoluta de l'intèrpret Python que Ansible farà servir a les víctimes. Aquesta opció és útil quan la màquina víctima no poseeix l'intèrpret Python a la ruta per defecte, que és "/usr/bin/python" (per exemple, a Fedora és /usr/bin/python3)

NOTA: Si es vol indicar un intèrpret Python concret per defecte per totes les màquines que puguin haver indicades a l'inventori, es pot optar per establir directament la directiva *interpreter_python=* sota la secció *[defaults]* de l'arxiu "ansible.cfg". En qualsevol cas, si no s'indica cap intèrpret específic de cap manera, Ansible intentarà trobar-ne un automàticament (https://docs.ansible.com/ansible/latest/reference_appendices/interpreter_discovery.html)

Todas estas opciones *ansible_** se pasarán luego como variables homónimas a cada playbook que se ejecute en las víctimas en cuestión.

Para no tener el inventario demasiado atiborrado de variables, no obstante, en vez de indicar las opciones *ansible_** y/o las variables personalizadas directamente dentro de él, lo que se suele hacer es indicar el nombre y valor de las distintas variables (con esta sintaxis: *nombre: valor*, y una variable por línia) en un fichero YAML con extensión ".yml" o ".yaml" llamado obligatoriamente o "**/etc/ansible/group_vars/nombreGrupoInvent.yml**" o "**/etc/ansible/host_vars/nombreMaqInvent.yml**", según si las variables en cuestión queremos que afecten a un grupo o a una "víctima" en particular (las carpetas "group_vars" y "host_vars" habrá que crearlas si no existen).

NOTA: Las carpetas "group_vars" y "host_vars" también pueden existir dentro del directorio donde esté ubicado el playbook en cuestión. En todo caso, si ambas rutas existen, las variables existentes bajo "/etc/ansible" serán sobreescritas por estas últimas.

Una misma variable puede estar indicada en varios lugares a la vez. A continuación se muestra el orden de precedencia de una misma variable según donde se encuentre definida (de menor a mayor importancia; cada lugar sobreescribe a todos sus anteriores; muchos de los lugares indicados se irán conociendo y estudiando a lo largo de este documento). En https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#variable-precedence-where-should-i-put-a-variable se encuentra la lista oficial de precedencia.

*Variables definidas para el grupo [*all*] en un inventario
*Variables definidas para un grupo padre en un inventario
*Variables definidas para un grupo hijo en un inventario
*Variables definidas para un host concreto en un inventario
*Facts obtenidos de un host
*Variables definidas en la sección "vars" de un play
*Variables definidas en la sección "vars_prompt" de un play
*Variables definidas en la sección "vars_files" de un play
*Variables definidas en la sección "block" de un play
*Variables definidas en una tarea concreta de un play
*Variables definidas mediante "include_vars"
*Variables definidas mediante "set_facts" o "registered_vars"
*Variables indicadas mediante el parámetro *-e* del intérprete de comandos

Uso directo de Ansible "ad-hoc" (sin "playbooks")

Las operaciones se pueden lanzar directamente mediante el comando *ansible* de la siguiente manera:

ansible nombreGrupo -m modulo [-a "param1=valor1 param2=valor2 ..."] -u usuario -k [-b -K]

donde:

*_nombreGrupo_ representa el nombre de un grupo indicado en "/etc/ansible/hosts" (podría ser también la palabra "*all*" o "*localhost*")

NOTA: También se pueden especificar varios grupos a la vez con la sintaxis *nomGrup1:nomGrup2:...*
Pero si se quiere indicar solo las máquinas que pertenezcan obligatoriamente a los dos grupos indicados, se puede escribir *nomGrup1:&nomGrup2* Si se quiere indicar solo las máquinas que pertenezcan al primer grupo excepto si pertenecen también al segundo, se puede escribir *nomGrup1:!nomGrup2* Para más información, consultar https://docs.ansible.com/ansible/latest/user_guide/intro_patterns.html

*_-m modulo_ representa el módulo concreto de Ansible a utilizar

*_-a "..."_: Si el módulo a utilizar necesitara parámetros para funcionar (algunos los requieren, otros no), se han de indicar como una lista de nombre<->valor separados por espacios

*_-u usuario_ representa el usuario de la "víctima" con el que se entrará en ella a realizar la operación. No es pot indicar usuaris diferents per "víctimes" diferents: totes les "víctimes" hauran de tenir el mateix nom d'usuari per poder ser controlades des d'Ansible.

*_-k_ : Hace que Ansible pida interactivamente la contraseña del usuario anterior (este parámetro es imprescindible si no hemos activado todavía la autentiación por claves o si no hemos indicado la contraseña en el inventario; si sí hemos hecho alguna de estas dos cosas, no se deberá escribir).

*_-b :_ Si la operación a realizar necesita ejecutarse con permisos de administrador, este parámetro indica que se ejecute el comando *sudo* automático. Por tanto, será imprescindible escribirlo para la mayoría de operaciones del sistema. Obviamente, si el usuario indicado no pudiera utilizar el comando *sudo* en la máquina "víctima", este parámetro no funcionará.

*_-K_ : Si en realizarse el *sudo*, este está configurado para solicitar contraseña interactivamente (su configuración por defecto es así), este parámetro permite introducirla (a no ser que la hayamos indicado explícitamente en el inventario).

*_-l unaIP,otraIP,..._ : Indica un subconjunto de IPs (o nombres) de víctimas (las cuales han de estar presentes en el inventario utilizado dentro de algún grupo mayor) sobre las que, y solo a ellas, se les aplicará el módulo indicado. Este parámetro es útil para no tener que explicitar un subconjunto de víctimas en el inventario si ya hay un conjunto mayor definido donde estas ya aparecen.

NOTA: También se pueden especificar rangos mediante corchetes (ya sea con IPs o con nombres, por ejemplo así: *nombreMaquina[5-15]*). Consulteu https://docs.ansible.com/ansible/latest/user_guide/intro_patterns.html
NOTA: Si no se está seguro de en qué hosts funcionará *--limit*, puede usar la opción *--list-hosts*. Al agregar *–list-hosts* se obtendrá una lista de hosts en los que eventualmente se ejecutaría el módulo.

*_-t /ruta/carpeta_ : Indica la carpeta donde se guardará, por cada máquina "víctima" probada, un fichero (cuyo nombre será la IP de la "víctima" correspondiente) conteniendo lo mismo que se muestra en pantalla relativo a esa "víctima" en concreto tras la ejecución del comando ansible

*_-f nº_ : Se puede indicar como parámetro también el nivel de paralelismo que se desea al lanzar las operaciones, sobreescribiendo así el valor de la directiva general *forks*

*_-c nombreConexion:_ Indica el tipo de conexión a establecer con las víctimas ("ssh", "local", etc). Si la víctima es 127.0.0.1 (o "localhost") automáticamente se elige conexión "local"; si no, por defecto es "ssh". Otros valores posibles son "podman", "lxd", etc

Ejemplos de uso de módulos básicos

A continuación veremos algunos ejemplos de módulos básicos (una lista más exhaustiva se puede encontrar en https://docs.ansible.com/ansible/latest/collections/ansible/builtin) :

*_ping_ : Ejecuta el comando ping a las "víctimas" indicadas: *ansible all -m ping -u usuari -k*

*_setup_ : Antes de ejecutar cualquier tarea, Ansible recopila automáticamente información sobre el sistema que está administrando. Esta información se devuelve a la máquina controladora en forma de un objeto JSON llamado "**ansible_facts**". Los elementos dentro de este objeto se denominan "facts" e incluyen una amplia gama de información del sistema, como el número de núcleos de CPU, direcciones ipv4 e ipv6, discos montados, distribución de Linux y más. Ansible usa estos datos para varios propósitos internos, pero podemos ver y guardar esta información gracias al módulo "setup". Este módulo muestra una lista de todos los datos disponibles sobre una máquina: *ansible all -m setup -u usuari -k*

NOTA: Los "facts" recopilados se pueden filtrar (para mostrar solo unos pocos) usando el parámetro *-a "filter=someFactName"* (donde el "fact" indicado, no obstante, solo puede estar en la subclave de primer nivel del objeto "ansible_facts" para poder ser filtrado; por otro lado, los comodines - * y? - pueden usarse para especificar su nombre).
NOTA: También está el parámetro *-a "gather_subset = someValue, anotherValue"*. Si se proporciona, restringe los "facts" recopilados al subconjunto dado. Valores posibles: "all" (por defecto), "min", "hardware", "network" y "virtual". Estos valores también se pueden utilizar con una inicial "!" para especificar que ese subconjunto concreto no debe recopilarse (por ejemplo: "*!hardware,!network*"). Hay que tener en cuenta que si se especifica "*!all*", se recopila igualmente el subconjunto *min*: para evitar recopilar incluso el subconjunto *min*, hay que especificar "*!all,!min*". De hecho, para recopilar solo "facts" específicos, es mejor usar "*!all,!min*" y entonces especificar los subconjuntos de "facts" particulares.

*_shell_ : Ejecuta en las "víctimas" indicadas un comando arbitrario especificado en el parámetro *-a*: *ansible all -m shell -a "free -m" -u usuari -k*

NOTA: Otro parámetro que ofrece este módulo interesante a conocer es *chdir=/ruta/carpeta*, el cual hace "moverse" a Ansible a la carpeta indicada antes de ejecutar el comando en cuestión. Para escribirlo (y, en general, todos los parámetros de este módulo) se ha de indicar a continuación del comando en sí, por ejemplo así: *ansible all -m shell -a "ls -l chdir=/var"*
NOTA: Son también interesantes los siguientes módulos similares:
* *script* (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/script_module.html)
* *command* (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/command_module.html)
* *expect* (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/expect_module.html)

*_apt_ o *_dnf_ : Instala/Desintala en las "víctimas" Ubuntu/Fedora indicadas el programa especificado en el parámetro *-a : ansible all -m apt -a "name=prog1 state=present " -u usuari -k -b -K* , donde *name* indica el paquete (o paquetes, separados por comas!) a instalar (también puede ser la ruta absoluta de un fichero ".deb"/".rpm"!) y *state* indica el estado final deseado (también puede ser "latest" para actualizarlo a la última versión o "absent" para desinstalarlo).

NOTA: También podemos actualizar todo el sistema simplemente indicando el parámetro *-a "upgrade=dist"* (si usamos el módulo *apt*) o los parámetros *-a "name='*' state=latest"* (si usamos el módulo *dnf*)
NOTA: El módulo *package* ofrece un subconjunto básico de los otros módulos (*apt, dnf,...)* pero tiene la ventaja de ser genérico, permitiéndonos no tener que saber de antemano el gestor de paquetes utilizado en las máquinas víctimas (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/package_module.html).

*_service_ : Controla en las "víctimas" indicadas un servicio especificado en el parámetro *-a*, así: *ansible all -m service -a "name=serv1 state=started" -u usuari -k -b -K* Además de parar/iniciar también puede habilitar/deshabilitar (*enabled=yes/no*), entre otras opciones.

NOTA: Este módulo es genérico para diferentes gestores de servicios existentes (Systemd, SysV, OpenRC, etc) pero si sabemos que nuestras máquinas "víctimas" usan todas Systemd, podemos usar como alternativa el módulo específico *systemd* (https://docs.ansible.com/ansible/latest/collections/ansible/builtin/systemd_module.html), que dispone de opciones muy similares

*_copy_ : Copia ficheros desde la máquina controladora hacia las "víctimas" indicadas. Mediante el parámetro *-a* se ha de indicar como mínimo qué fichero/s se quieren copiar (parámetro *src*) y a qué carpeta de destino (parámetro *dest*), indicando opcionalmente el propietario y permisos que tendrán los ficheros copiados: *ansible all -m copy -a "src=/ruta/fich dest=/etc/nuevonombrefich owner=root group=root mode=0640" -u usuari -k*

> NOTA: La ruta local del archivo a copiar a las "víctimas" puede ser absoluta o relativa. Esta ruta, por otro lado, también puede ser un directorio; si es así, se copia de forma recursiva. En este caso, si la ruta termina con "/", solo los contenidos internos de ese directorio se copian en el destino. De lo contrario, si no termina con "/", se copia el directorio con todo el contenido.
> NOTA: En vez de *src* se puede usar el parámetro *content="algo"* para indicar un determinado contenido textual a grabar en el fichero destino (similar a ejecutar *echo "algo" > fichero.dest* en la máquina "víctima").

*_file_ : Este módulo permite realizar diversas acciones sobre un determinado fichero remoto de las "víctimas" indicadas (cuya ruta se especifica con el parámetro *path)* según el valor del parámetro *state*. Por ejemplo:

-a "path=/etc/fstab state=touch" (crea fichero vacío)
-a "path=/etc/fstab.lnk src=/etc/fstab state=link" (crea link)
-a "path=/opt/public state=directory" (crea carpeta)
-a "path=/etc/fstab state=absent" (borra fichero -o carpeta recursivamente-)
-a "path=/opt state=directory mode=775 owner=ana group=ana recurse=yes" (cambia propietario y permisos de fichero o carpeta)

*_user_ : Administra usuarios en las "víctimas". Dispone de varios parámetros importantes como *name* (el nombre del usuario), *password* (la contraseña encriptada...para encriptarla deberemos ejecutar primero el comando *openssl passwd -6 contrasenya*; este parámetro solo se escribirá cuando se quiera crear el usuario o modificar su contraseña), *groups* (para indicar los grupos donde se añadirá el usuario, bien en su creación o a posteriori), *append* (para indicar si los grupos indicados con el parámetro anterior son los únicos -si vale *no*- o se añadirán a los ya existentes -si vale *yes*-), *shell* (para indicar/modificar el shell por defecto del usuario), *password_lock* (para, si vale "yes", bloquear la contraseña del usuario en cuestión -ya existente- y, si vale "no", desbloquear-la), etc.

> NOTA: Si quisiéramos eliminar un usuario, deberíamos utilizar simplemente los parámetros *name* y *state=absent* (y opcionamente, el parámetro *remove=yes* para borrar también la carpeta personal del usuario a eliminar). Por defecto, el valor del parámetro *state* es "present", por lo que normalmente no se indicará al no ser necesario
> NOTA: Atenció! A la contrasenya encriptada obtinguda de la comanda *openssl* indicada al paràgraf anterior apareixen símbols "$". Aquests símbols s'han d'escapar (és a dir, afegir una "\" al davant) en el moment d'escriure-la com a valor del paràmetre "password" del mòdul "user" (si no la contrasenya no s'assignarà correctament). És a dir, si representa que la contrasenya encriptada és 6qwb23$2ghw , el paràmetre "password" s'haurà d'escriure de la següent manera: password=\$6\$qwb23\$2ghw
> NOTA: Si en assignar el valor del paràmetre "password" d'aquest mòdul no volem "copiar-pegar" manualment el hash obtingut de la comanda *openssl* , una opció és escriure directament això: password=$(openssl passwd -6 contrasenya). Noteu, però, que aquest "truc" no el podrem fer servir dins de "playbooks" (el que estudiarem a continuació) perquè es basa en una sintaxi coneguda pel Bash (els símbols $(...)) però a l'hora d'interpretar el contingut escrit dins dels "playbooks" el Bash no hi intervé

Todos los módulos anteriores (y la gran mayoría de los que veremos a continuación) tienen un nombre "canónico" formado por el prefijo "***ansible.builtin.***" más su nombre corto. Es decir, que en realidad, el nombre del módulo "ping" es *ansible.builtin.ping*, el del módulo "setup" es *ansible.builtin.setup*, y así. Esto es debido a que Ansible está estructurado en forma de colecciones de módulos para que solamente tengamos que tener instaladas las colecciones que necesitemos realmente (si no fuera así, tendríamos centeneras de módulos instalados inútiles). Sobre la gestión de las colecciones y cómo instalarlas, desinstalarlas, etc hablaremos más adelante, pero lo que ahora basta por saber es que, solo por el hecho de estar instalado, Ansible ya incorpora por defecto un conjunto de módulos predefinidos básicos que siempre estarán, independientemente de si se instalan más colecciones o no. A todos esos módulos predefinidos se les identifica bajo el prefijo "*ansible.builtin*". El hecho de que exista un prefijo es para evitar que un eventual módulo de alguna eventual colección tuviera el mismo nombre que el de otro módulo de otra colección que hubiéramos instalado: utilizando prefijos como espacios de nombres, se evita ese posible problema. Si no existe ambigüedad, no obstante, ese prefijo se puede omitir por comodidad (que es lo que haremos).

Uso de "playbooks"

Para realizar más de una tarea de forma conjunta (y secuencial), se han de crear "playbooks", los cuales son simplemente archivos de texto escritos en YAML (y, por ello, con extensión *.yml) que contienen una lista de elementos (llamados "plays") que definen las operaciones a realizar, cómo, dónde y por quién. En concreto, en un "play" podemos distinguir dos partes:

*Una parte genérica, donde básicamente se establecen los hosts sobre los cuales se realizarán las tareas, mediante qué usuario se realizarán y las eventuales variables de entorno necesarias para ello, entre otras configuraciones generales

*La lista concreta de tareas que representan las operaciones deseadas a ejecutar en los hosts indicados en la parte genérica (y por el usuario también indicado allí). Esta lista es un array YAML.

En teoría, en un "playbook" podría haber más de un "play", escrito uno tras otro. Sin embargo, esto es poco común: lo más habitual es escribir "playbooks" formados por un solo "play". En cualquier caso, cada "play" (aunque solo haya uno) ha de indicarse como un elemento de un array YAML.

En la parte genérica de un "play" nos podemos encontrar con los siguientes elementos:

*_name_ : Nombre del play (ha de ser descriptivo sobre las tareas que realizará)

*_hosts_ : Nombre del grupo de hosts (definido en el inventario utilizado, el cual puede ser , como ya sabemos, el general o el indicado mediante el parámetro *-i* o la variable de entorno **ANSIBLE_INVENTORY**) al que se le aplicarán las operaciones

*_remote_user_ : Usuario de la "víctima" con el que se entrará en ella a realizar las operaciones. Si no se indica, se deberá seguir utilizando el parámetro *-u* en el terminal a la hora de ejecutar el "playbook" (mediante el comando *ansible-playbook*, como ahora veremos). Igualmente, si usamos autenticación por contraseña (o no la hemos indicado en el inventario), será necesario indicar también el parámetro *-k* en el terminal porque no existe ninguna directiva para indicar la contraseña SSH dentro de un playbook.

*_become_: Si vale *yes*, es equivalente a indicar el parámetro *-b* del comando *ansible* . Si no hemos indicado la contraseña *sudo* en el inventario (o no hemos configurado previamente *sudo* para no pedir contraseña) deberemos indicar obligatoriamente el parámetro *-K* a mano a la hora de ejecutar el "playbook" (mediante el comando *ansible-playbook*, como ahora veremos).

*_vars_ : Opcional. Array que representa el conjunto de variables cuyos valores podrán utilizar las operaciones definidas a lo largo del "playbook". Para definir el array la sintaxis es:

```
vars:
    nombreVar1:  valor1
    nombreVar2:  valor2
```

Para utilizar el valor de una variable determinada en cualquier sitio posterior del "playbook", la sintaxis es: *{{ nombreVar1 }}*

> NOTA: Como ya sabemos, las variables que se hayan definido en un inventario o ficheros "group_vars"/"host_vars" equivalente se podrán utilizar en cualquier lugar de un playbook directamente. Igualmente ocurre con los "facts" recogidos. Consultar la tabla de precedencia mostrada en páginas anteriores para saber qué ubicación sobreescribe a qué otra.

*_gather_facts_ : Opcional (por defecto vale *yes*). Si vale *no*, Ansible no realizará una recogida previa de metadatos de las máquinas víctimas, los cuales contienen información sobre su CPU, su RAM, sus discos duros, sus interfícies y configuración de red, etc y pueden ser útiles -o no- para las tareas a realizar. Estos metadatos son los llamados "facts" y es la misma información que obtiene el módulo *setup* explícitamente.

NOTA: El valor por defecto de la directiva "gather_facts" es "yes" porque en el fichero "/etc/ansible/ansible.cfg" existe la directiva "gathering" con el valor "implicit". Si esta directiva valiera "explicit", el valor por defecto sería "no" y deberíamos explicitar la línia "gather_facts: yes" en los playbooks donde quisiéramos obtener los "facts".

La lista de tareas a ejecutar se compone, por otra parte, de los siguientes elementos:

*_tasks_ : Array YAML que representa el conjunto de operaciones a realizar EN ORDEN (a no ser que se use el ítem *order* con un valor diferente del por defecto, que es *"sorted"*...ya lo veremos). Cada una de ellas contendrá a su vez los siguientes ítems básicamente:

*_name: Descripción de la tarea_
*_nombreModulo : param1=valor1 param2=valor2 ..._
*_ignore_errors: yes_ (opcional)
*_become: yes_ (opcional)

NOTA: La línea *ignore_errors* por defecto vale *no*; en ese caso, la ejecución del "playbook" se para para un determinado host remoto al detectar algún error en la realización de una tarea en ese host. Per tenir un control molt més detallat sobre com s'ha de comportar Ansible en trobar errors durant l'execució d'un playbook que el que ofereix l'opció *ignore_errors*, hi ha diferents solucions que estudiarem més endavant (*failed_when, changed_when, meta:clear_host_errors, any_errors_fatal:true, max_fail_percentage* ...); en qualsevol cas, totes elles es poden consultar a https://docs.ansible.com/ansible/latest/user_guide/playbooks_error_handling.html amb exemples.
NOTA: La línea *become* es útil si no se desea utilizar la opción homónima para todo el "playbook" porque solo se necesita para una tarea en concreto.
NOTA: En comptes d'escriure-les totes seguides a la mateixa línia, les parelles nom<->valor dels mòduls indicats es poden escriure cadascuna en una línia diferent (tabulades respecte el nom del mòdul en qüestió, i a la mateixa alçada) fent servir el caràcters ":" de separador en comptes del "=". És a dir:

```
nombreModulo:
        param1: valor1
        param2: valor2
        ...
```

Por ejemplo, un "playbook" sencillo (con un solo "play" que realiza tres tareas) podría ser este:

```
- name: Hacer ping, instalar gnuchess e iniciar Apache2 en Fedora para unGrupoDelInventario
  hosts: unGrupoDelInventario
  remote_user: pepito
  tasks:
  - name: Hacer ping
    ping:
  - name: Instalar gnuchess en Fedora
    dnf: name=gnuchess state=latest
    become: yes
  - name: Iniciar Apache2 en Fedora
    systemd: name=httpd2 state=started
    become: yes
```

Una vez definido el "playbook", para ejecutarlo se deberá escribir el siguiente comando:

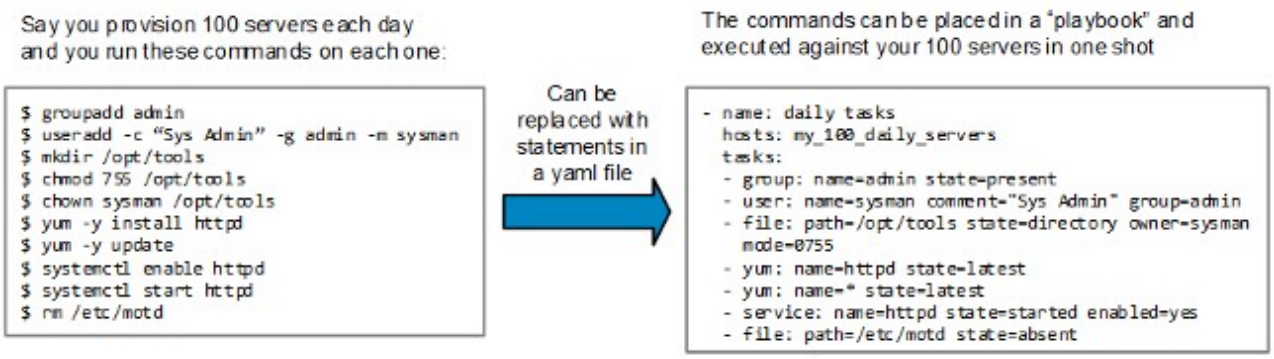
ansible-playbook nombrePlaybook.yml [-k][-K]

NOTA: Es posible pasar valores desde el terminal a variables definidas en el array "vars" del playbook indicando el parámetro *-e "variable1=valor1 variable2=valor2 ..."* en el comando *ansible-playbook* y definiendo el array "vars" así:

```
vars:
    nombreVarEnPlaybook: "{{ nombreVarPasadoPorTerminal }}"
    nombreOtraVarEnPlaybook: " {{ nombreOtraVarPasadoPorTerminal }}"
    nombreOtraVarMásEnPlaybook : unValorExplícito
```

También se pueden especificar valores de variables en ficheros exclusivos aparte, con la misma sintaxis que el contenido del array "vars", y utilizar esos valores en un "playbook" indicando la ruta del/los fichero/s en cuestión como valor/es de un ítem general llamado *vars_files* o, de forma similar, mediante el módulo *include_vars*

Un ejemplo gráfico de la utilidad de los "playbooks" es el siguiente diagrama:



Truc per fer que *sudo* no pregunti contrasenya (i així no haver de fer servir el paràmetre -K)

Si l'usuari que fem servir per entrar amb l'Ansible a les màquines "víctima" ja pot fer *sudo* en elles (només que introduïnt la contrasenya), l'únic que haurem de fer serà modificar la següent línia de l'arxiu "/etc/sudoers" de cadascuna d'aquestes víctimes...:

%sudo ALL=(ALL:ALL) ALL

...per

%sudo ALL=(ALL:ALL) ***NOPASSWD:*** *ALL*

NOTA: A Fedora, aquesta línia no existeix però n'hi ha una altra d'equivalent que comença per *%wheel*. Caldrà fer el mateix canvi

Si l'usuari que fem servir per entrar amb l'Ansible a les màquines "víctima" no pot fer *sudo* en elles, haurem de fer el mateix pas anterior però, a més, haurem d'executar la següent comanda: *sudo usermod -aG sudo nomusuari* (a Fedora seria *sudo usermod -aG wheel nomusuari*)

Truc per fer que SSH no pregunti contrasenya (i així no haver de fer servir el paràmetre -k)

La forma més efectiva de què un servidor SSH (en aquest cas, el corresponent a cada "víctima") no pregunti contrasenya és fent servir el mecanisme d'autenticació per claus. Els passos necessaris per fer això són:

1.-Crea, a la màquina controladora, i essent el mateix usuari que executarà la comanda ansible, el parell de claus SSH executant la comanda: *ssh-keygen*

2.-Copia el contingut de la clau pública recentment creada (suposarem que és l'arxiu "~/.ssh/id_rsa.pub") dins de l'arxiu "/home/pepito/.ssh/authorized_keys" de cadascuna de les "víctimes". Això es pot aconseguir de vàries formes (via e-mail, pendrive, *scp*...o, si es connecta amb les víctimes amb el mateix usuari que executarà la comanda ansible, així):

ssh-copy-id pepito@ipVictima1
ssh-copy-id pepito@ipVictima2
...

NOTA: Opcional. Para deshabilitar el acceso por contraseña (y, por tanto, conseguir así que solo se pueda acceder a las máquinas "víctimas" solo si se intenta acceder desde la máquina controladora -y solamente siendo el usuario "usuari"-), en la configuración del servidor SSH de la máquina imagen (fichero "/etc/ssh/sshd_config") hay que poner "no" en vez de "yes" en la línea PasswordAuthentication (es decir, dejarla así: ***PasswordAuthentication no***) y ya está; el resto de líneas relevantes ya están bien como están. (*PermitRootLogin prohibit-password* y *PubKeyAuthentication yes*) . Una vez hecho esto, reinicia el servicio SSH

EXERCICIS:

0.-a) Crea (o reusa) dues màquines virtuals de tipus Server amb la seva tarja de xarxa en mode "adaptador pont", amb 1GB de RAM i que tinguin totes dues un usuari anomenat "pepito" que pugui fer *sudo*

> NOTA: Recorda que per crear un usuari i que pugui fer *sudo* cal que executis les següents comandes:
> *A Ubuntu: *sudo useradd -m -G sudo pepito && sudo passwd pepito*
> *A Fedora: *sudo useradd -m -G wheel pepito && sudo passwd pepito*

aII) En aquestes dues màquines (que anomenarem "B" i "C" respectivament) hauràs d'instal.lar els paquets "openssh-server" i "python3" i comprovar que el dimoni *sshd* estigui encés (i el tallafocs aturat).

b) Crea (o reutilitza) una altra màquina virtual Fedora Server també amb la seva tarja de xarxa en mode "adaptador pont" i 1 GB de RAM . En aquesta màquina (l'anomenarem "A") hauràs d'instal.lar els paquets "ansible-core" i "sshpass" (aquest darrer és necessari per a què Ansible pugui demanar interactivament les contrasenyes dels usuaris per accedir via SSH). Aquesta màquina serà la que farem servir per administrar les màquines "B" i "C".

c) Edita la llista de hosts remots reconeguts per l'Ansible (el que es diu el seu "inventari") per tal d'afegir un nou grup anomenat *[victimes]* que inclogui les IPs de "B" i "C".

d) Crea l'arxiu "/etc/ansible/ansible.cfg" amb la configuració per defecte de l'Ansible amb totes les seves directives comentades (pista: repassa l'ús de la comanda *ansible-config* als paràgrafs de teoria) i tot seguit modifica la directiva adient per tal de no realitzar la comprovació de hosts en el moment de connectar-se a màquines remotes.

> NOTA: Si aquest darrer pas no es realitza, hi hauria una altra manera d'evitar l'error que Ansible mostra però és molt més pesat: consistiria en entrar prèviament una a una a totes les màquines remotes amb el client SSH estàndar per tal de respondre "yes" a la pregunta que sempre apareix la primera vegada que hom es conecta a un host remot

1.-a) Utilitza el mòdul "**dnf**" d'Ansible amb la comanda *ansible* per tal d'instal.lar a aquestes dues màquines a la vegada, mitjançant l'usuari comú ("pepito"), els paquets "socat" i "nmap". Comprova seguidament que hi siguin (provant-les d'executar a cadascuna d'elles "in-situ", per exemple).

> NOTA: En el cas de què les màquines "víctima" fossin Ubuntu, s'hauria d'emprar el mòdul "**apt**"

aII) Torna a repetir l'apartat anterior i comprova si la sortida mostrada per la comanda *ansible* és diferent. ¿Per què?

b) Utilitza el mòdul "dnf" d'Ansible amb la comanda *ansible* per desinstal.lar a les màquines "B" i "C" només el paquet "socat". Comprova seguidament que ja no hi sigui (provant-la d'executar a cadascuna d'elles "in-situ", per exemple).

c) Utilitza el mòdul "dnf" d'Ansible amb la comanda *ansible* per tal d'actualitzar tots els paquets però només de la màquina "B"

2.-a) Utilitza el mòdul "**user**" d'Ansible amb la comanda *ansible* per tal de crear a les màquines "B" i "C" un nou usuari anomenat "manolo" amb contrasenya "1234" que pertanyi només al grup "adm" i que tingui el shell /bin/bash. Comprova que hi sigui (ho pots provar fent *su -l manolo* a les màquines administrades i després *id*)

b) Utilitza el mòdul "user" d'Ansible amb la comanda *ansible* per afegir aquest usuari "manolo", un cop creat, al grup "sudo". Comprova seguidament que hi pertanyi (a més de la resta de grups als que ja pertanyia); això ho pots fer executant *su -l manolo* a les màquines administrades i després *id*)

c) Bloqueja l'usuari "manolo" (de forma similar a com ho faria la comanda *usermod -L*) però només a la màquina "B". Prova-ho i comprova-ho intentant iniciar sessió "in-situ" en la màquina "B" (no podràs; pots comprovar a més que, efectivament apareix el símbol "!" al davant del seu "hash" en l'arxiu "/etc/shadow") i en la màquina "C" (si podràs)

d) Utilitza el mòdul "user" d'Ansible amb la comanda *ansible* per eliminar aquest usuari "manolo" de totes les màquines, incloent l'esborrament de la seva carpeta personal

3.-a) Modifica el valor de "ansible_ssh_pass" pel grup *[victimes]* a l'inventari de l'Ansible per tal de què SSH no demani cap contrasenya. Seguidament utilitza la comanda *ansible* per, per exemple, realitzar un ping a les màquines "B" i "C" (mitjançant el mòdul "ping") i comprovar que, efectivament, no necessites introduïr cap contrasenya (és a dir, que no cal que escriguis el paràmetre *-k*)

b) Desfés el que has realitzat a l'apartat anterior (que no deixa de ser un "nyap"). Segueix ara les passes indicades a l'últim quadre de la teoria per aconseguir que el servidor SSH de les màquines "B" i "C" no demani contrasenya. Tot seguit utilitza la comanda *ansible* per, per exemple, realitzar un ping a les màquines "B" i "C" (mitjançant el mòdul "ping") i comprovar que tampoc no necessites introduïr cap contrasenya.

Tot i haver realitzat el pas anterior, si haguéssim d'executar algun mòdul que necessités "escalar privilegis" amb *sudo* hauríem de seguir escrivint una contrasenya (en aquest cas la del *sudo*). Per evitar això podries o bé establir la directiva "ansible_become_pass" a l'inventori de hosts (de forma semblant a l'apartat a)) o bé seguir les passes manuals indicades a la teoria per modificar l'arxiu "/etc/sudoers" de cada víctima. No obstant, per no haver de repetir els mateixos pasos una vegada rera l'altra a cada màquina "víctima" de forma manual, farem la modificació dels seus arxius "/etc/sudoers" (de fet, d'una sola línia d'aquests arxius) d'una forma alternativa, fent servir precisament la comanda *ansible*, que per això està: per automatitzar tasques a múltiples màquines remotes. Concretament:

c) A partir de consultar la documentació oficial del mòdul "**lineinfile**" (disponible aquí: https://docs.ansible.com/ansible/latest/collections/ansible/builtin/lineinfile_module.html) digues què faria la comanda següent (i executa-la): *ansible victimes -m lineinfile -a "path=/etc/sudoers state=present regexp='^%wheel' line='%wheel ALL=(ALL) NOPASSWD: ALL'" -u pepito -b -K*

NOTA: Si les màquines-víctima són Ubuntu en comptes de Fedora, canvia *%wheel* per *%sudo* a la comanda anterior

d) Un cop realitzat l'apartat anterior, utilitza ara el mòdul "**shell**" d'Ansible per executar la comanda *fdisk -l* a les màquines "B" i "C" com a administrador: hauràs de veure que ara hauràs d'indicar igualment el paràmetre *-b* però no pas el paràmetre *-K* (perquè no necessitaràs introduïr cap contrasenya).

e) ¿Què fa la comanda *ansible victimes -m shell -a "ls -l /etc | grep '^d'| wc -l" -u usuari* ?

4.-a) Utilitza el mòdul "**file**" d'Ansible amb la comanda *ansible* per crear a les màquines "B" i "C" la carpeta "/home/pepito/prova" (amb permisos 755 i propietari "usuari")

b) Utilitza el mòdul "file" d'Ansible amb la comanda *ansible* per canviar, a les mateixes màquines "B" i "C", els permisos de la carpeta "/home/pepito/prova" recentment creada per a què ara siguin 700.

c) Utilitza el mateix mòdul "file" per crear, a les mateixes màquines "B" i "C", l'arxiu buit "/home/pepito/prova/hola.txt" (amb els permisos i propietari per defecte)

d) Repeteix l'apartat anterior però ara afegint el paràmetre *-b* a la comanda *ansible* executada. ¿Quin és el propietari ara del fitxer creat? ¿Per què?

e) Utilitza el mateix mòdul "file" per crear, a les mateixes màquines "B" i "C", un enllaç anomenat "/home/pepito/enllaç_a_hola.txt" que apunti a l'arxiu creat a l'apartat anterior.

f) Després de confirmar que tots els elements creats als apartats anteriors existeixin, utilitza de nou el mòdul "file" per esborrar-los tots (enllaç, fitxer i carpeta).

> NOTA: Hauràs d'executar les esborrades una per una: el mòdul "file" no accepta comodins. Per poder fer-ho de cop caldria recórrer a l'ús de bucles en playbooks, els quals els estudiarem properament o bé utilitzar altres mòduls com ara "shell" o un altre que estudiarem també properament, "find", entre altres possibilitats.

5.-a) Utilitza el mòdul "**copy**" d'Ansible amb la comanda *ansible* per copiar l'arxiu "/etc/nsswitch.conf" de la màquina "A" a les màquines "B" i "C" amb el nom de "$HOME/suich.conf"

b) Utilitza el mòdul "**setup**" d'Ansible amb la comanda *ansible* i, amb l'ajuda de *grep, cut, jq...* escriu un petit shell script que mostri <u>només</u> la quantitat total de memòria RAM que tenen les màquines "B" i "C".

> NOTA: Com saber quina clau concreta necessiteu aillar, d'entre les tantes que hi ha, una ajuda pot ser observar la sortida de *ansible victimes -u pepito -m setup | sed "1c {" | jq ".ansible_facts|keys"*

c) Utilitza el mòdul "**systemd**" (o "service") d'Ansible amb la comanda *ansible* per tal d'aturar i també deshabilitar el servei "ufw" (a Ubuntu) o "firewalld" (a Fedora) de les màquines "B" i "C".

> NOTA: Si volguéssim saber simplement l'estat d'un servei (és a dir, si està aturat o no), el mòdul "systemd" (o "service") no ens serveixen. Caldria utilitzar el mòdul "service_facts", que estudiarem properament

6.-a) Escriu (i executa) un "playbook" que faci el mateix que l'apartat a) de l'exercici 1. Comprova que el paquets "socat" i "nmap" tornin a estar instal.lats a les màquines "B" i "C".

b) Escriu (i executa) un "playbook" que faci el mateix que l'apartat a) de l'exercici 2. Comprova que l'usuari "manolo" torni a aparèixer a les màquines "B" i "C". ¿Com modificaries el "playbook" per a què en comptes de crear només l'usuari "manolo" creés a més l'usuari "luisa" i "ana"?

c) Escriu (i executa) un "playbook" que faci el mateix que els apartat a), c) i e) de l'exercici 4. Comprova que, efectivament, tornin a aparèixer a les màquines "B" i "C".