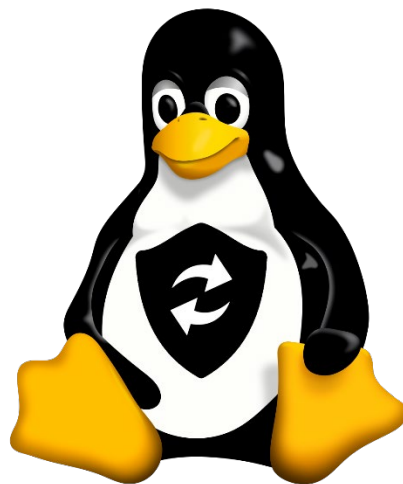




GESTIÓN Y AUTOMATIZACIÓN DE LAS ACTUALIZACIONES DE SEGURIDAD EN LINUX



Álvaro Saavedra de la Peña Úbeda

Fecha de realización: mayo 2021

1. AGRADECIMIENTOS

A la empresa Everis donde he realizado las practicas este curso 2020/2021 por su gran ayuda con respecto a facilitarme documentación, libros, videos explicativos sobre Ansible, Docker, Linux, Gitlab, etc. Que he podido aplicar en este trabajo de fin de grado.

Y a mi tutor académico Luis Antolín que me ha estado guiando y asesorando en la elaboración de este trabajo.

ÍNDICE

Contenido

1. AGRADECIMIENTOS.....	- 1 -
2. RESUMEN	- 4 -
3. ANTECEDENTES / INTRODUCCIÓN	- 5 -
4. OBJETIVOS GENERALES, ESPECÍFICOS Y ALCANCE DEL PROYECTO.....	- 6 -
4.1. Objetivo general	- 6 -
4.2. Objetivos específicos	- 6 -
4.3. Alcance del proyecto	- 6 -
5. DEFINICIONES	- 7 -
6. NOTACIONES Y SIMBOLOS	- 9 -
7. DESARROLLO DEL TRABAJO FINAL	- 10 -
7.1. Script de Ubuntu 18.04.....	- 11 -
7.2. Instalación de Ansible AWX	- 15 -
7.2.1. Requisitos para instalar Ansible AWX	- 15 -
7.2.1.1. ¿Qué es Docker?	- 16 -
7.2.1.2. ¿Qué es Docker - compose?	- 17 -
7.2.1.3. ¿Qué es Git?.....	- 18 -
7.2. Configuración de AWX.....	- 22 -
7.2.2. Añadir máquinas al servidor AWX:	- 24 -
7.2.1.4. ¿Qué es el servicio SSH?	- 24 -
7.3. Ejecución de los scripts	- 29 -
7.3.2. ¿Cómo ejecutar los scripts en los equipos?	- 29 -
7.3.3. Lanzando los scripts	- 32 -
7.4. Resultado final del script.....	- 33 -
7.4.2. Actualizar los paquetes listados:	- 34 -
8. CONCLUSIONES Y RECOMENDACIONES	- 37 -
8.2. Proyección a futuro:	- 37 -
9. REFERENCIAS.....	- 38 -
10. BIBLIOGRAFÍA.....	- 39 -

11.	ANEXOS.....	- 40 -
-----	-------------	--------

2. RESUMEN

2.1. Resumen del proyecto

Muchas empresas cuando quieren actualizar los paquetes de seguridad de su servidor o máquinas quieren saber cuáles se van a actualizar y los cambios que estos van a producir, para no comprometer la seguridad de la empresa y ver la compatibilidad con el hardware que tienen.

Para obtener los paquetes de seguridad que se van a actualizar voy a realizar unos scripts que listen dichos paquetes con diferente información para valorar si se quieren actualizar o no. Estos van a estar alojados en un repositorio de GitHub creado por mí y a través de la herramienta *Ansible* en *AWX* los ejecutaremos. Como servidor para el *AWX* voy a utilizar Ubuntu 20.04 y los clientes sobre los que voy a ejecutar los scripts son dos máquinas con Ubuntu 18.04 y dos máquinas con Debian 10.

En definitiva, este proyecto se basa en dar una solución a un problema que muchas empresas tienen.

2.2. Summary of the project

Many companies when they want to update the security packages of their server or machines want to know the packages that are going to be updated and the changes that these are going to produce in order not to compromise the security of the company and see the compatibility with their hardware.

To obtain the security packages that are going to be updated I am going to make some scripts that list these packages with different information to evaluate if they want to be updated. These are going to be hosted in a GitHub repository created by me and through the Ansible tool in AWX we will run them. As server for the AWX I will use an Ubuntu 20.04 and the clients on which I will run the scripts are two machines with Ubuntu 18.04 and two machines with Debian 10.

Definitely, this project is based on giving a solution to a problem that many companies have.

3. ANTECEDENTES / INTRODUCCIÓN

3.1. ¿Por qué utilizar Ansible para ejecutar unos scripts?

Ansible es una herramienta de software que sirve para automatizar el despliegue de software, configuraciones y aplicaciones en muchos equipos al mismo tiempo. Para el administrador de software de una empresa es muy cómodo utilizarlo, ya que le ahorra mucho tiempo puesto que no tiene que dedicarle a cada equipo horas hasta tener la configuración deseada en cada uno de ellos.

Su configuración e instalación en los ordenadores donde se vaya a ejecutar es muy básica, únicamente se necesita Python.

Es muy fácil aprender a usarlo ya que utiliza YAML como lenguaje y no se necesitan excesivos conocimientos de programación para su uso.

3.2. Ansible AWX

Ansible simplemente es el método que utilizamos para lanzar esas configuraciones que queramos mandar. Necesitamos una herramienta que nos permita configurar sobre qué equipos queremos actuar y para ello utilizaremos AWX.

AWX tiene una interfaz web muy sencilla y cómoda de utilizar. Es un proyecto de código abierto creado por Red Hat que permite a los usuarios controlar mejor el uso de Ansible en entornos de TI.

Red Hat tiene dos vertientes de AWX, la versión open source que es AWX y su versión comercial AWX Tower. En este trabajo se desarrollará la utilización de la vertiente AWX ya que no tiene ningún coste adicional y es accesible a todo el mundo.

4. OBJETIVOS GENERALES, ESPECÍFICOS Y ALCANCE DEL PROYECTO

4.1. Objetivo general

Listar las actualizaciones de seguridad de varias distribuciones de Linux mediante unos scripts y utilizar Ansible AWX para ejecutarlos.

4.2. Objetivos específicos

- Conocer el funcionamiento de Ansible para poder expresarlo.
- Utilizar un sistema Linux para el desarrollo de todo el proyecto.
- Comenzar a crear mi propio portfolio como profesional técnico y almacenarlo en GitHub.

4.3. Alcance del proyecto

Publicar mi repositorio de GitHub donde se alojan todos los scripts y documentación necesaria para que empresas que necesiten este servicio lo puedan utilizar.

5. DEFINICIONES

Ansible: Es un software que automatiza el aprovisionamiento de software, la gestión de configuraciones y el despliegue de aplicaciones. Está categorizado como una herramienta de orquestación, muy útil para los administradores de sistema y DevOps.

AWX: Es un proyecto comunitario de código abierto que proporciona software para administrar proyectos de Ansible. AWX se aloja en GitHub y proporciona una interfaz de usuario basada en la web.

Bash: Es un intérprete de comandos que ejecuta, una por una, las instrucciones introducidas por el usuario o contenidas en un script y devuelve los resultados. Actúa como interfaz entre el kernel Linux y los usuarios o programas del modo texto.

CVE: Es una lista de información registrada sobre vulnerabilidades de seguridad conocidas, en la que cada referencia tiene un número de identificación CVE-ID, descripción de la vulnerabilidad, que versiones del software están afectadas, posible solución al fallo.

DevOps: Es un método de desarrollo de software que se centra en la colaboración, comunicación e integración entre los ingenieros de sistemas y los desarrolladores de software.

Docker: Es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

Docker Compose: A diferencia de Docker, Docker Compose es una herramienta desarrollada para ayudar a definir y compartir aplicaciones de varios contenedores simultáneamente.

ECS: Es un servicio de administración de contenedores muy escalable y rápido que facilita la ejecución, detención y administración de contenedores en un clúster.

GitHub: Es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador. Está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no sólo puedas

descargarte la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo.

Repositorio: Un repositorio es una lista de programas, siempre actualizada, que nos permite buscar y descargar fácilmente todo tipo de programas y herramientas de nuestra distribución de Linux.

Script: Se trata de un código de programación, que contiene comandos u ordenes que se van ejecutando de manera secuencial

YAML: es un formato de serialización de datos legible por humanos inspirado en lenguajes como XML, C, Python, Perl, así como el formato para correos electrónicos.

6. NOTACIONES Y SIMBOLOS

CVE: Common Vulnerabilities and Exposures.

ECS: Elastic Container Service.

DevOps: Development y operations.

7. DESARROLLO DEL TRABAJO FINAL

Una empresa cuando necesita un listado de los paquetes que se van a actualizar no solamente quiere el nombre del paquete y el número de la nueva versión.

A continuación, se desarrollan los datos que también se necesitan obtener:

Urgencia

La urgencia de un paquete es la prioridad con la que se debe actualizar.

Hay tres tipos de urgencias, medium, high y low.

		Urgency		
		High	Medium	Low
Impact	High	Critical	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Low

CVE

El CVE (*Common Vulnerabilities and Exposures*) es un número asociado a cada versión de un paquete que nos muestra una descripción de la vulnerabilidad, si ya tiene solución en caso de fallo, etc.

Por ejemplo, el paquete *tcpdump* en su versión 4.9.2-3 tiene el siguiente CVE *CVE-2017-16808*. Si buscamos dicho código en la siguiente página https://cve.mitre.org/cve/search_cve_list.html podemos obtener información del paquete *tcpdump*.

Search Results

There are 1 CVE Records that match your search.	
Name	Description
CVE-2017-16808	tcpdump before 4.9.3 has a heap-based buffer over-read related to aoe_print in print-aoe.c and lookup_emem in addrtoname.c.

Repositorio

Otro dato que las empresas solicitan es el repositorio del que se descarga el paquete a actualizar. Hay dos tipos de repositorios hoy en día, los oficiales y los no oficiales. Muchas veces introducimos nuevos repositorios a nuestro sistema y queremos ver de cuál se va a descargar el paquete para saber si es de confianza. El repositorio es el servidor donde se alojan todas las aplicaciones a descargar en nuestro sistema Linux.

Release date

Release date o fecha de lanzamiento es la fecha en la cual ha sido publicada la actualización del paquete. Este dato nos es muy útil para saber cuánto tiempo lleva lanzada esa versión.

Todos esos datos los vamos a obtener mediante un script.

7.1. Script de Ubuntu 18.04¹

Los scripts están realizados en bash. Es una shell de Unix, y un intérprete de lenguaje de comandos. Una shell es simplemente un macroprocesador que ejecuta comandos. Es la shell más utilizada de forma predeterminada para la mayoría de las distribuciones de Linux.



Para obtener los datos mencionados en el apartado anterior se han obtenido de la siguiente manera:

Urgencia

```
apt-get changelog $pkg | grep 'urgency' 2>/dev/null | head -n 1 |  
awk -F = '{ print $2 }'
```

Con el comando `changelog` accedemos a un archivo el cual contiene un registro de los cambios que se realizarán tras la actualización del paquete que le indiquemos.

¹ Ver apartado REFERENCIAS o clicar en el título "Script de Ubuntu 18.04" para ver el código.

La urgencia la podemos ver en la segunda línea columna 59. Para filtrar y obtener únicamente la palabra "medium" que es lo que nos interesa filtramos con `grep`, imprimimos únicamente la primera opción que nos aparezca y a eso le hacemos un:

```
awk -F = '{ print $2 }'
```

```
Get:1 https://changelogs.ubuntu.com tcpdump 4.9.3-0ubuntu0.18.04.1 Changelog [33,9 kB]
tcpdump (4.9.3-0ubuntu0.18.04.1) bionic-security; urgency=medium

* SECURITY UPDATE: Updated to 4.9.3 to fix multiple security issues
- debian/patches/disable-tests.diff: disable tests that require newer
  libpcap.
- CVE-2017-16808, CVE-2018-10103, CVE-2018-10105, CVE-2018-14461,
  CVE-2018-14462, CVE-2018-14463, CVE-2018-14464, CVE-2018-14465,
  CVE-2018-14466, CVE-2018-14467, CVE-2018-14468, CVE-2018-14469,
  CVE-2018-14470, CVE-2018-14879, CVE-2018-14880, CVE-2018-14881,
  CVE-2018-14882, CVE-2018-16227, CVE-2018-16228, CVE-2018-16229,
  CVE-2018-16230, CVE-2018-16300, CVE-2018-16451, CVE-2018-16452,
  CVE-2018-19519, CVE-2019-1010220, CVE-2019-15166, CVE-2019-15167

-- Marc Deslauriers <marc.deslauriers@ubuntu.com> Fri, 24 Jan 2020 07:57:54 -0500
```

CVE

```
apt changelog $pkg 2>/dev/null | awk '/^Get:/{flag=1} flag{buf = buf
$0 ORS} /^ -- /{flag=0; if(imprimir=1) print buf; buf="";
imprimir=0} flag && /CVE/{imprimir=1}' | egrep CVE-20[0-2][0-9]-[0-
9]*[,]?$ | awk '{ for (i=1; i<=NF; i++) print $i }' | tr -d ',' |
tail -n+2
```

Para obtener el CVE del paquete volvemos a acceder al `changelog` y en este caso filtramos con `awk` desde la palabra `Get:` hasta `-` y cogemos todos los casos en los que aparezca la palabra `CVE`.

```
Get:1 https://changelogs.ubuntu.com tcpdump 4.9.3-0ubuntu0.18.04.1 Changelog [33,9 kB]
tcpdump (4.9.3-0ubuntu0.18.04.1) bionic-security; urgency=medium

* SECURITY UPDATE: Updated to 4.9.3 to fix multiple security issues
- debian/patches/disable-tests.diff: disable tests that require newer
  libpcap.
- CVE-2017-16808, CVE-2018-10103, CVE-2018-10105, CVE-2018-14461,
  CVE-2018-14462, CVE-2018-14463, CVE-2018-14464, CVE-2018-14465,
  CVE-2018-14466, CVE-2018-14467, CVE-2018-14468, CVE-2018-14469,
  CVE-2018-14470, CVE-2018-14879, CVE-2018-14880, CVE-2018-14881,
  CVE-2018-14882, CVE-2018-16227, CVE-2018-16228, CVE-2018-16229,
  CVE-2018-16230, CVE-2018-16300, CVE-2018-16451, CVE-2018-16452,
  CVE-2018-19519, CVE-2019-1010220, CVE-2019-15166, CVE-2019-15167

-- Marc Deslauriers <marc.deslauriers@ubuntu.com> Fri, 24 Jan 2020 07:57:54 -0500
```

Repositorio

```
apt-cache policy tcpdump | awk '/Installed/{flag=1} flag{buf = buf $0 ORS} /Packages/{flag=0; if(imprimir=1) print buf; buf=""; imprimir=0} flag && /http/{imprimir=1}' | grep http | awk '{print $2}'
```

En el caso del repositorio el comando es el mostrado baajo. De igual manera filtramos con `awk` en este caso buscamos la palabra `Installed` y el primer enlace que se encuentre que empiece por `http`:

```
alvaro@alvaro-VirtualBox:~$ apt-cache policy tcpdump
tcpdump:
  Installed: 4.9.3-0ubuntu0.18.04.1
  Candidate: 4.9.3-0ubuntu0.18.04.1
  Version table:
*** 4.9.3-0ubuntu0.18.04.1 500
    500 http://es.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packa
ges
    500 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packag
es
    100 /var/lib/dpkg/status
    4.9.2-3 500
    500 http://es.archive.ubuntu.com/ubuntu bionic/main amd64 Packages
alvaro@alvaro-VirtualBox:~$
```

Release date

```
apt changelog $pkg 2>/dev/null | awk '/^Get:/{flag=1} flag{buf = buf $0 ORS} /^ --/{flag=0; if(imprimir=1) print buf; buf=""; imprimir=0}flag && /CVE/{imprimir=1}' | egrep [A-Z][a-z][a-z], | awk -F "," '{ print $2 }' | awk '{print $3"-"$2"-"$1}' | sed 's/Jan/01/g; s/Feb/02/g; s/Mar/03/g; s/Apr/04/g; s/May/05/g ; s/Jun/06/g; s/Jul/07/g; s/Aug/08/g; s/Sep/09/g; s/Oct/10/g; s/Nov/11/g; s/Dec/12/g'
```

Para el release date acedemos al `changelog` de nuevo y filtramos de nuevo con `awk` hasta conseguir obtener la fecha con un formato estandarizado como puede ser el siguiente: `2020-01-24`

```
Get:1 https://changelogs.ubuntu.com tcpdump 4.9.3-0ubuntu0.18.04.1 Changelog [3
3,9 kB]
tcpdump (4.9.3-0ubuntu0.18.04.1) bionic-security; urgency=medium

* SECURITY UPDATE: Updated to 4.9.3 to fix multiple security issues
- debian/patches/disable-tests.diff: disable tests that require newer
  libpcap.
- CVE-2017-16808, CVE-2018-10103, CVE-2018-10105, CVE-2018-14461,
  CVE-2018-14462, CVE-2018-14463, CVE-2018-14464, CVE-2018-14465,
  CVE-2018-14466, CVE-2018-14467, CVE-2018-14468, CVE-2018-14469,
  CVE-2018-14470, CVE-2018-14879, CVE-2018-14880, CVE-2018-14881,
  CVE-2018-14882, CVE-2018-16227, CVE-2018-16228, CVE-2018-16229,
  CVE-2018-16230, CVE-2018-16300, CVE-2018-16451, CVE-2018-16452,
  CVE-2018-19519, CVE-2019-1010220, CVE-2019-15166, CVE-2019-15167

-- Marc Deslauriers <marc.deslauriers@ubuntu.com> Fri, 24 Jan 2020 07:57:54 -
0500
```

Datos adicionales que se añaden son el hostname del equipo y la fecha en la que se ha lanzado el script. El nombre del equipo lo obtenemos con el comando `hostname` y para la fecha en la que se ha ejecutado el script utilizamos el siguiente comando `ls /tmp/ --full-time | grep security_updates.json | awk '{print $6,$7}' | awk -F. '{print $1}'`.

En caso de que no haya actualizaciones de seguridad se mostrará el siguiente código con el mensaje “There are no packages with security updates”.

```
1 {
2   "repositories":[
3     {
4       "repository": "",
5       "packages": [],
6       "metadata": {
7         "status": "No packages to update",
8         "reason_status": "There are no packages with security updates"
9       }
10    }
11  ],
12  "metadata": {
13    "machine-hostname": "${hostname}",
14    "generation_data_date": "$(ls /tmp/ --full-time | grep security_updates.json | awk '{print $6,$7}' | awk -F. '{print $1}')"
15  }
16 }
```

Esos son los datos principales que se obtienen con el script.

Todos los datos obtenidos se almacenan en variables las cuales se acaban imprimiendo un archivo con formato .json .

```
1 urg+=( $(apt-get changelog $pkg | grep 'urgency' 2>/dev/null | head -n 1 | awk -F = '{ print $2 }') )
2 rls+=( $(apt changelog $pkg 2>/dev/null | awk '/^Get:/{flag=1} flag{buf = buf $0 ORS} /^ -- /{flag=0;
3 repositories+=( $(apt-cache policy $pkg | awk '/Installed/{flag=1} flag{buf = buf $0 ORS} /Packages/{
4 arr+=( $(apt-cache policy $pkg | awk '/Installed/{flag=1} flag{buf = buf $0 ORS} /Packages/{flag=0; if
5 apt changelog $pkg 2>/dev/null | awk '/^Get:/{flag=1} flag{buf = buf $0 ORS} /^ -- /{flag=0; if(imprin
```

El porqué de exportarlo en este formato .json es porque es muy legible y fácil de comprender para su lectura.

7.2. Instalación de Ansible AWX

Para instalar Ansible AWX necesitamos un servidor Linux o Windows, yo he preferido trabajar con Linux.

Todas las máquinas que se usan en este trabajo están virtualizadas mediante VirtualBox, es un software de virtualización con el cual hemos trabajado en clase.



Concretamente he decidido utilizar un servidor Ubuntu 20.04. El porqué de este y no otras distribuciones de Linux es porque hay mucha más documentación de AWX en Ubuntu que en CentOS por ejemplo. También por la comodidad que me genera trabajar con Ubuntu ya que es una distribución que hemos utilizado mucho más en clase y estoy más familiarizado con ella.

7.2.1. Requisitos para instalar Ansible AWX

Para la instalación de AWX necesitamos tener una máquina de por lo menos los siguientes requisitos:

- 4GB de memoria RAM
- 2 núcleos de procesador
- 10GB de disco duro

En mi caso tengo los siguientes requisitos:

- 4GB de memoria RAM
- 2 núcleos de procesador
- 20GB de disco duro

```
alvaro@alvarosv:~$ neofetch
  .-/+00SSSS00+/- .
  `:+SSSSSSSSSSSSSSSSSS+:`
  -+SSSSSSSSSSSSSSSSSSyySSSS+-
  .0SSSSSSSSSSSSSSSSSSdMMMMNySSSS0.
  /SSSSSSSSSSSShdmmNNmmyNMMMMHSSSSS/
  +SSSSSSSSSShmydMMMMMMMMNdddySSSSSSS+
  /SSSSSSSShNMMMyhhyyyhNMMMNhSSSSSSS/
  .SSSSSSSSdMMMNhSSSSSSSShNMMMdSSSSSSS.
  +SSShhhyNMMNySSSSSSSSSSyNMMMySSSSSS+
  0SSyNMMMNyMMHSSSSSSSSSSShmmhSSSSSS0
  0SSyNMMMNyMMHSSSSSSSSSSShmmhSSSSSS0
  +SSShhhyNMMNySSSSSSSSSSyNMMMySSSSSS+
  .SSSSSSSSdMMMNhSSSSSSSShNMMMdSSSSSSS.
  /SSSSSSSShNMMMyhhyyyhNMMMNhSSSSSSS/
  +SSSSSSSSdmydMMMMMMMMNdddySSSSSSS+
  /SSSSSSSSSShdmmNNmmyNMMMMHSSSSS/
  .0SSSSSSSSSSSSSSSSSSdMMMMNySSSS0.
  -+SSSSSSSSSSSSSSSSSSyySSSS+-
  `:+SSSSSSSSSSSSSSSSSS+:`
  .-/+00SSSS00+/- .

alvaro@alvarosv
-----
OS: Ubuntu 20.04.2 LTS x86_64
Host: VirtualBox 1.2
Kernel: 5.4.0-72-generic
Uptime: 3 mins
Packages: 1736 (dpkg), 4 (snap)
Shell: bash 5.0.17
Resolution: 1920x975
DE: GNOME
WM: Mutter
WM Theme: Adwaita
Theme: Yaru [GTK2/3]
Icons: Yaru [GTK2/3]
Terminal: gnome-terminal
CPU: AMD Ryzen 5 3400G (2) @ 3.699GHz
GPU: 00:02.0 VMware SVGA II Adapter
Memory: 1997MiB / 3936MiB
```


Las herramientas que necesitamos instalar para poder instalar el servidor AWX son las siguientes:

Siempre antes de instalar cualquier programa en Linux tenemos que actualizar los repositorios con:

```
sudo apt update
```

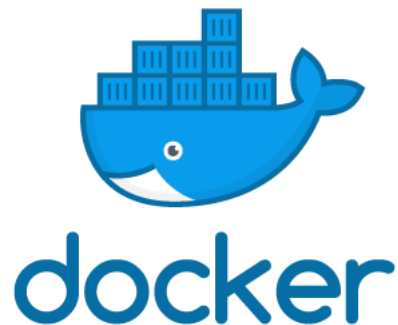
Una vez actualizados los repositorios podemos instalar Ansible con el siguiente comando:

```
sudo apt install ansible
```

Ansible AWX se va a ejecutar en varios contenedores de Docker con lo cual necesitamos tener Docker y Docker - compose instalado.

7.2.1.1. ¿Qué es Docker?

Docker es una herramienta que sirve para crear contenedores ligeros y portables en los cuales poder instalar aplicaciones y servicios de software. Estos pueden ejecutarse en cualquier máquina con Docker instalado, no importa el sistema operativo que la máquina tenga por debajo.



¿Docker vs VM?

La diferencia con una máquina virtual es el tamaño que esta necesita, una máquina virtual necesita tener un sistema operativo y en el instalar todos los programas y servicios que queramos. En cambio para que un contenedor Docker funcione necesitas únicamente un sistema operativo con Docker instalado y a partir de ahí levantas el servicio o programa que necesitas. Dicho contenedor le puedes exportar y compartir. Un contenedor ocupa megas de espacio y una máquina virtual ocupa GB.

La configuración para levantar servicios en Docker se realiza mediante un archivo llamado "Dockerfile".

Este es el ejemplo de un archivo Dockerfile que levanta un Ubuntu 18.04 y le instala Python a la hora de ejecutarse.

```
1 #Example simple Dockerfile
2 FROM ubuntu:18.04
3 COPY . /app
4 RUN make /app
5 CMD python /app/app.py
```



La instalación de Docker es muy sencilla solo hay que seguir estos pasos:
Importamos la clave GPG del repositorio oficial de Docker:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Y añadimos el repositorio a nuestra máquina Ubuntu:

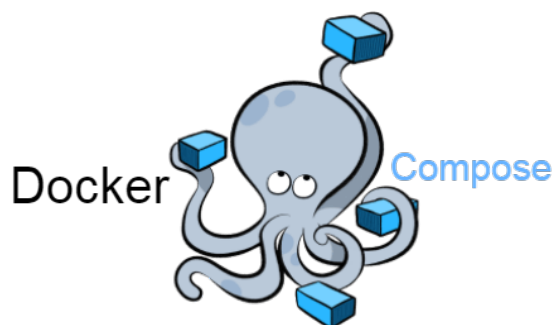
```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

Actualizamos el repositorio nuevo y podemos instalar Docker con:

```
sudo apt update && sudo apt install docker-ce docker-ce-cli
```

7.2.1.2. ¿Qué es Docker - compose?

Con Docker funciona muy bien el levantar uno o dos contenedores simultáneamente, pero si necesitamos tener varios contenedores a la vez como es el caso del servidor AWX necesitamos utilizar Docker - compose ya que levanta cuatro contenedores simultáneamente.



```
alvaro@alvarosv:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
8a7e47a0a170   ansible/awx:15.0.1  "/usr/bin/tini -- /u..." 3 weeks ago   Up 2 minutes  8052/tcp                awx_task
6c611d814342   ansible/awx:15.0.1  "/usr/bin/tini -- /b..." 3 weeks ago   Up 2 minutes  0.0.0.0:80->8052/tcp, :::80->8052/tcp  awx_web
154e804796b9   redis           "docker-entrypoint.s..." 3 weeks ago   Up 2 minutes  6379/tcp                awx_redis
4cfe88c60993   postgres:10      "docker-entrypoint.s..." 3 weeks ago   Up 2 minutes  5432/tcp                awx_postgres
alvaro@alvarosv:~$
```

La configuración de Docker – compose se realiza mediante un archivo .yaml .

Para instalar Docker – compose insertamos el siguiente comando en la terminal:

```
alvaro@alvarosv:~/Documents$ curl -s https://api.github.com/repos/docker/compose/releases/latest \
> | grep browser_download_url \
> | grep docker-compose-Linux-x86_64 \
> | cut -d '"' -f 4 \
> | wget -qi -
alvaro@alvarosv:~/Documents$ ls -l
total 12436
-rw-rw-r-- 1 alvaro alvaro 12728384 Apr 14 17:08 docker-compose-Linux-x86_64
-rw-rw-r-- 1 alvaro alvaro      94 Apr 14 17:08 docker-compose-Linux-x86_64.sha256
alvaro@alvarosv:~/Documents$
```

Hacemos el archivo descargado ejecutable cambiándole los permisos:

```
chmod 0760 docker-compose-linux-x86_64
```

Y movemos el archivo a la ruta de Docker-compose:

```
sudo mv docker-compose-linux-x86_64 /usr/local/bin/docker-compose
```

Ya tenemos instalado Docker y Docker – compose. Para comprobar que están instalados introducimos los siguientes comandos:

```
alvaro@alvarosv:~$ docker --version
Docker version 20.10.6, build 370c289
alvaro@alvarosv:~$ docker-compose --version
docker-compose version 1.29.1, build unknown
alvaro@alvarosv:~$
```

Otro requisito que necesitamos es tener instalado git para poder clonarnos el repositorio de AWX. Y también Python3-pip.

7.2.1.3. ¿Qué es Git?

Git es una aplicación que sirve para subir código a internet. Con él se puede tener un control de versiones, nos va a servir para trabajar en equipo de una manera mucho más simple y optima cuando estamos desarrollando software.



Con Git vamos a poder controlar todos los cambios que se hacen en nuestra aplicación y en nuestro código y vamos a tener control absoluto de todo lo que pasa en el código, pudiendo volver atrás en el tiempo, y con la posibilidad de tener diferentes ramas de desarrollo, etc.

Como usuarios podemos clonarnos repositorios de git que la gente tenga públicos, este es el caso de Ansible AWX, todo su código lo tienen en su repositorio de Git y es público.

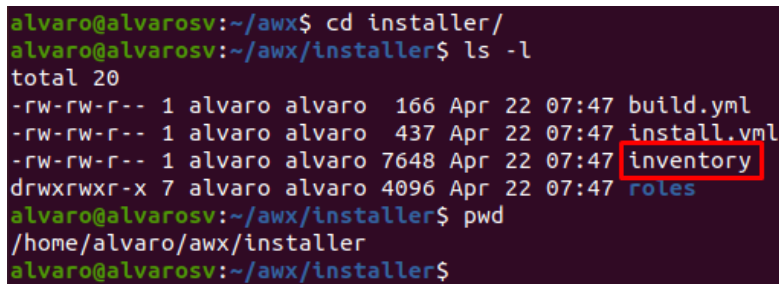
```
sudo apt install git python3-pip -y
```

Una vez tenemos todas esas herramientas instaladas comenzamos la instalación de AWX.

Nos clonamos el repositorio oficial de AWX.

```
git clone -b 15.0.1 https://github.com/ansible/awx.git
```

En este repositorio vamos a navegar hasta el archivo "inventory" que está en awx/installer.



```
alvaro@alvarosv:~/awx$ cd installer/
alvaro@alvarosv:~/awx/installer$ ls -l
total 20
-rw-rw-r-- 1 alvaro alvaro 166 Apr 22 07:47 build.yml
-rw-rw-r-- 1 alvaro alvaro 437 Apr 22 07:47 install.yml
-rw-rw-r-- 1 alvaro alvaro 7648 Apr 22 07:47 inventory
drwxrwxr-x 7 alvaro alvaro 4096 Apr 22 07:47 roles
alvaro@alvarosv:~/awx/installer$ pwd
/home/alvaro/awx/installer
alvaro@alvarosv:~/awx/installer$
```

En este fichero se definen todas las configuraciones del servidor que se va a levantar en el siguiente paso cuando lancemos el playbook. Como el límite de cpu que queremos destinar al servidor, los puertos a utilizar, el usuario y contraseña para acceder, una secret key que sirve para encriptar y tiene que ser única, etc.

Una vez configurado lo que queremos se nos quedaría un fichero así:

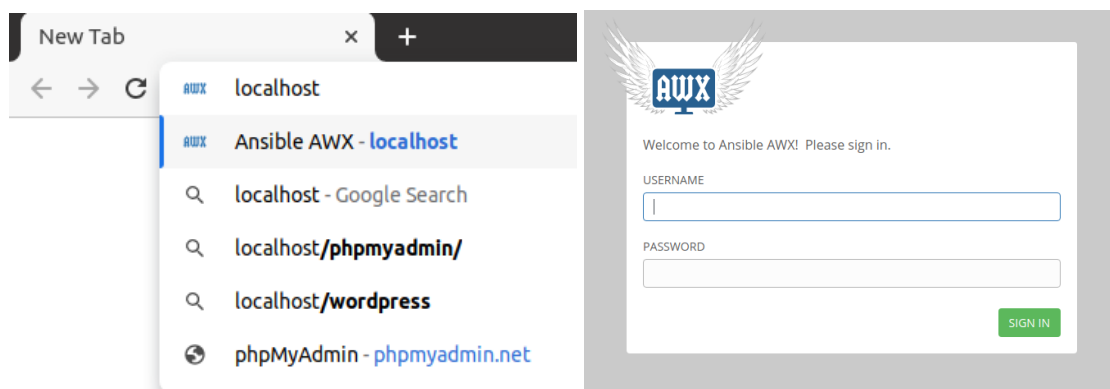
```
alvaro@alvarosv: ~/awx/installer
1 localhost ansible_connection=local ansible_python_interpreter="/usr/bin/env python3"
2
3 [all:vars]
4 dockerhub_base=ansible
5 awx_task_hostname=awx
6 awx_web_hostname=awxweb
7 postgres_data_dir=~/.awx/pgdocker"
8 host_port=80
9 host_port_ssl=443
10 docker_compose_dir=~/.awx/awxcompose"
11 pg_username=awx
12 pg_password=awxpass
13 pg_database=awx
14 pg_port=5432
15 admin_user=admin
16 admin_password=
17 create_preload_data=True
18 secret_key=
```

Para iniciar la instalación tenemos que introducir el siguiente comando:

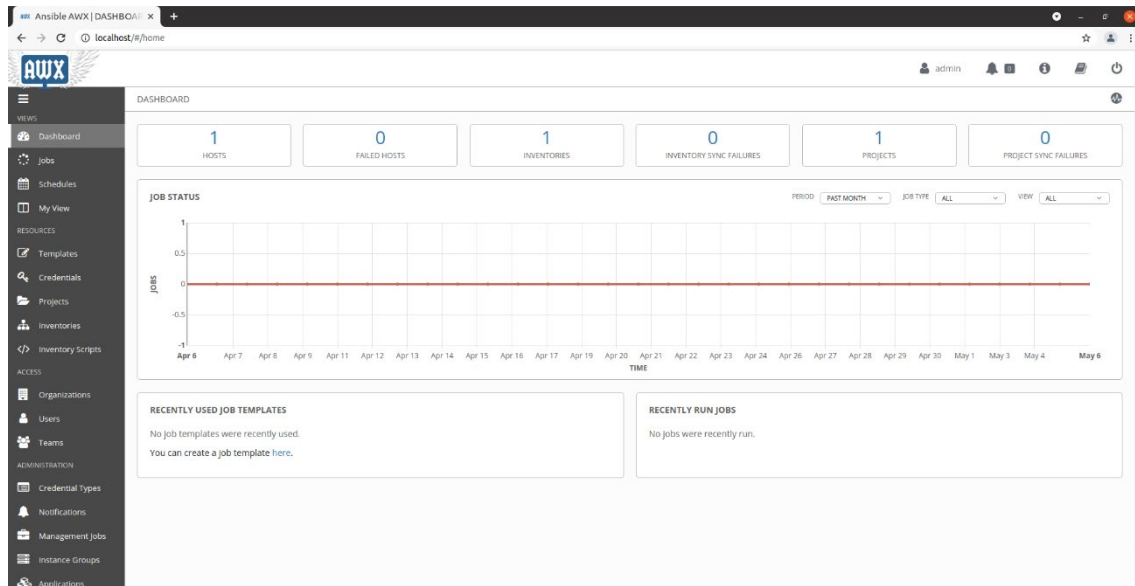
```
ansible-playbook inventory install.yml
```

Comenzará la instalación del servidor AWX, este proceso de levantar los contenedores y desplegarlos puede tardar hasta 5 minutos.

Cuando finaliza la instalación accedemos a nuestro navegador y nos conectamos a nuestra dirección localhost que es donde le hemos configurado que se levante.



Para acceder al servidor introducimos las credenciales que hemos puesto en el archivo de configuración y estamos dentro de nuestro servidor AWX.



7.2. Configuración de AWX

En este entorno web encontramos una barra lateral de navegación para acceder a las diferentes opciones.

Lo primero que tenemos que hacer es enlazar nuestro GitHub donde tenemos los scripts que se lanzarán sobre las máquinas. Para ello primero necesitamos añadir nuestro usuario y contraseña de GitHub. Nos vamos al apartado de credentials.

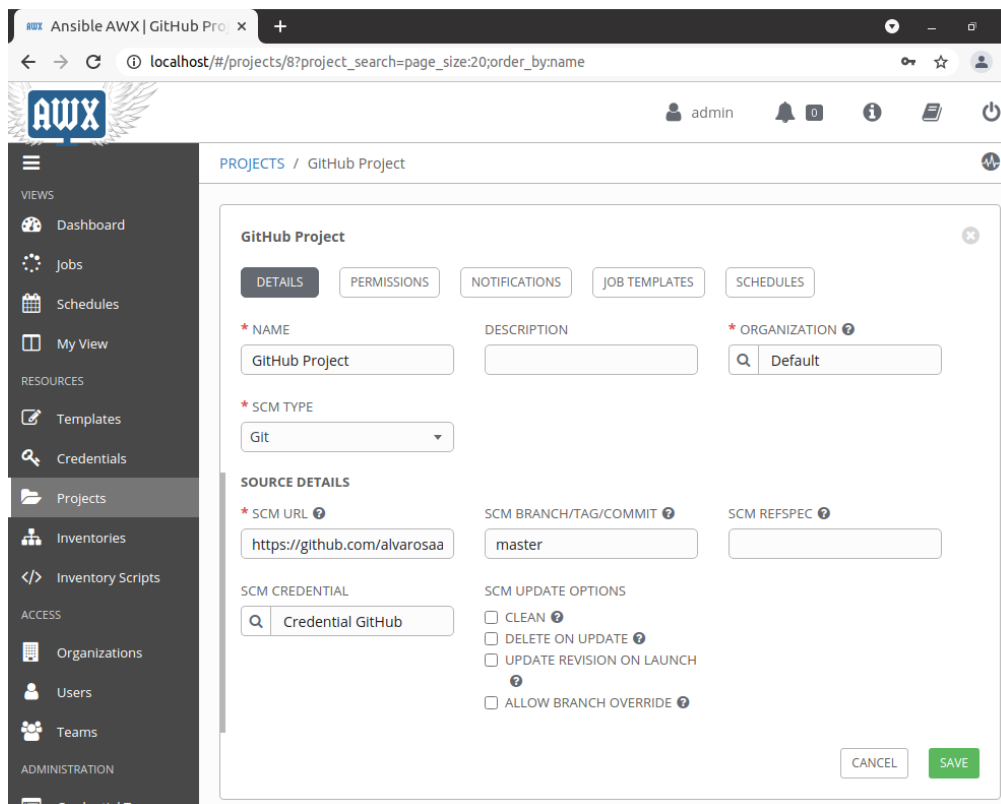


Ahí añadiremos una nueva credencial, encontramos muchos tipos de credenciales, la explicación de para que se utiliza cada una de ellas la encontramos en la documentación oficial de [Ansible](#)². En este caso utilizaremos la de "source control" nos pide nuestro usuario y contraseña de GitHub. También podemos configurar la contraseña mediante ssh.

² Ver apartado de REFERENCIAS o clicar en "Ansible".

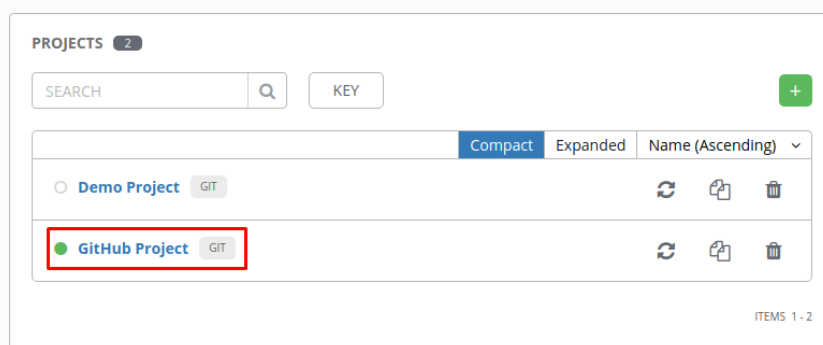
Para añadir nuestro repositorio de GitHub nos vamos al apartado de Projects, añadimos uno de tipo Git, en SCM URL introducimos nuestra url del repositorio y le decimos en que rama queremos que busque, en mi caso master. En SCM Credential seleccionamos la credencial de nuestro usuario de Git que hemos creado antes. Guardamos los cambios y empezará a buscar el repositorio y comprobar que coincide con las credenciales.

Si tuviésemos el repositorio en público no haría falta hacer el primer paso de crear la credencial ya que se conectaría sin contraseña.



The screenshot shows the 'GitHub Project' configuration page in the AWX interface. The left sidebar contains navigation links for Views (Dashboard, Jobs, Schedules, My View), Resources (Templates, Credentials, Projects, Inventories, Inventory Scripts), Access (Organizations, Users, Teams), and Administration (Credential Types). The main content area is titled 'PROJECTS / GitHub Project' and has tabs for DETAILS, PERMISSIONS, NOTIFICATIONS, JOB TEMPLATES, and SCHEDULES. The 'DETAILS' tab is active, showing fields for NAME (GitHub Project), DESCRIPTION, ORGANIZATION (Default), SCM TYPE (Git), SCM URL (https://github.com/alvarosaa), SCM BRANCH/TAG/COMMIT (master), and SCM REFSPEC. There is also a section for SCM CREDENTIAL (Credential GitHub) and SCM UPDATE OPTIONS (CLEAN, DELETE ON UPDATE, UPDATE REVISION ON LAUNCH, ALLOW BRANCH OVERRIDE). At the bottom right are 'CANCEL' and 'SAVE' buttons.

Cuando se consigue sincronizar nos aparece el punto verde que significa que lo ha encontrado y tenemos acceso a el.



The screenshot shows the 'PROJECTS' list in the AWX interface. The list has a search bar, a 'KEY' button, and a '+ ' button. The table has columns for 'Name (Ascending)' and 'GIT'. The 'GitHub Project' is listed with a green status dot, indicating it is synchronized. The 'Demo Project' is listed with a grey status dot. The table is in 'Compact' view. At the bottom right, it says 'ITEMS 1 - 2'.

Name (Ascending)	GIT
Demo Project	Grey dot
GitHub Project	Green dot

En este paso el servidor AWX ya es capaz de leer lo que tenemos dentro de nuestro repositorio de GitHub.

7.2.2. Añadir máquinas al servidor AWX:

¿Qué deben tener instalado las máquinas sobre las que queremos actuar?

Los hosts sobre los que vamos a lanzar los scripts únicamente necesitan tener instalado el servicio SSH que es por el cual se va a conectar el servidor al host.

En las distribuciones de Linux este servicio ya viene instalado por defecto, para comprobar si lo tenemos instalado escribimos el siguiente comando:

```
alvaro@alvaro-1:~$ ssh -V
OpenSSH_7.6p1 Ubuntu-4ubuntu0.3, OpenSSL 1.0.2n  7 Dec 2017
alvaro@alvaro-1:~$
```

En caso de que nos devuelva que no está instalado lo instalamos con el siguiente comando:

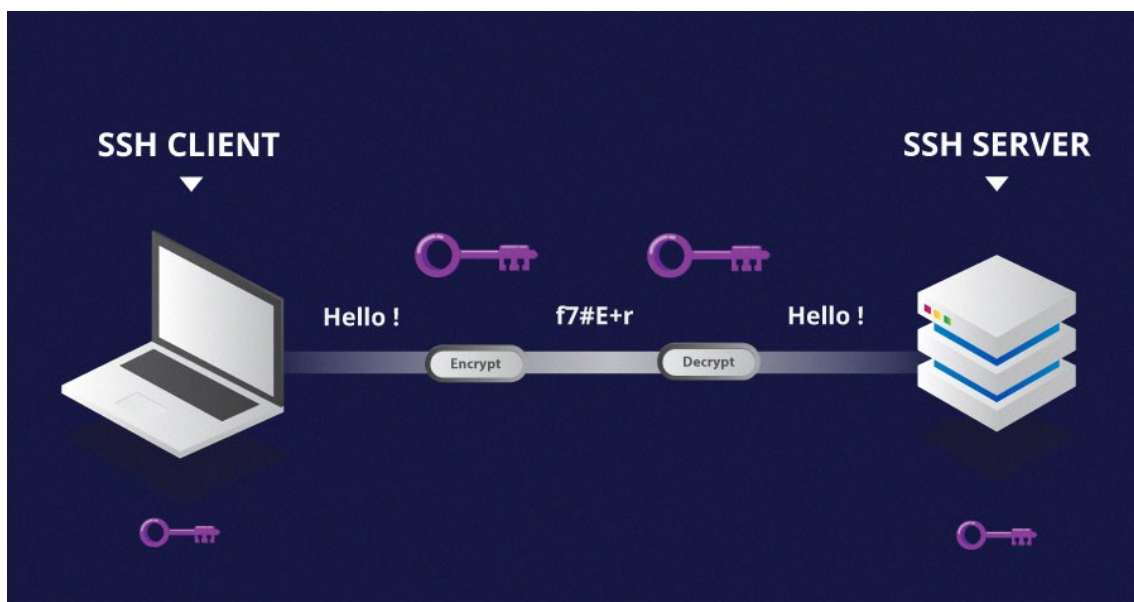
```
sudo apt install ssh -y
```

Este servicio tiene que estar siempre disponible cada vez que se encienda el ordenador. Para ello utilizamos el siguiente comando:

```
systemctl enable ssh
```

7.2.1.4. ¿Qué es el servicio SSH?

El servicio SSH es un protocolo de administración que permite conectarnos y controlar un ordenador de manera remota. La información que viaja a través de los equipos está cifrada. A diferencia del servicio FTP o TELNET que no lo está.



Para añadir los equipos contra los que vamos a lanzar los scripts necesitamos añadir la credencial de dichos equipos, al igual que con el usuario de GitHub tenemos que ir al apartado de “credentials” y creamos una contraseña por cada máquina que añadamos.

El tipo de credencial es “Machine” y nos pide nombre de usuario (debe ser un usuario con privilegios de administrador) y contraseña de este. La contraseña al igual que con el GitHub la podemos insertar mediante ssh.

The image displays two screenshots of the Ansible AWX web interface, specifically the 'EDIT CREDENTIAL' page. Both screenshots show the same form configuration for a credential named 'Ubuntu 18.04'.

Screenshot 1 (Top): The browser address bar shows the URL `localhost/#/credentials/4?credential_search=page_size:20;order_by:name`. The credential is titled 'Ubuntu 18.04 (1)'. The form fields are as follows:

- NAME:** Ubuntu 18.04 (1)
- DESCRIPTION:** Credential machine Ubu
- ORGANIZATION:** Default
- CREDENTIAL TYPE:** Machine
- TYPE DETAILS:**
 - USERNAME:** alvaro
 - PASSWORD:** (field with 'ENCRYPT' button and 'Prompt on launch' checkbox)

Screenshot 2 (Bottom): The browser address bar shows the URL `localhost/#/credentials/5`. The credential is titled 'Ubuntu 18.04 (2)'. The form fields are identical to the first screenshot:

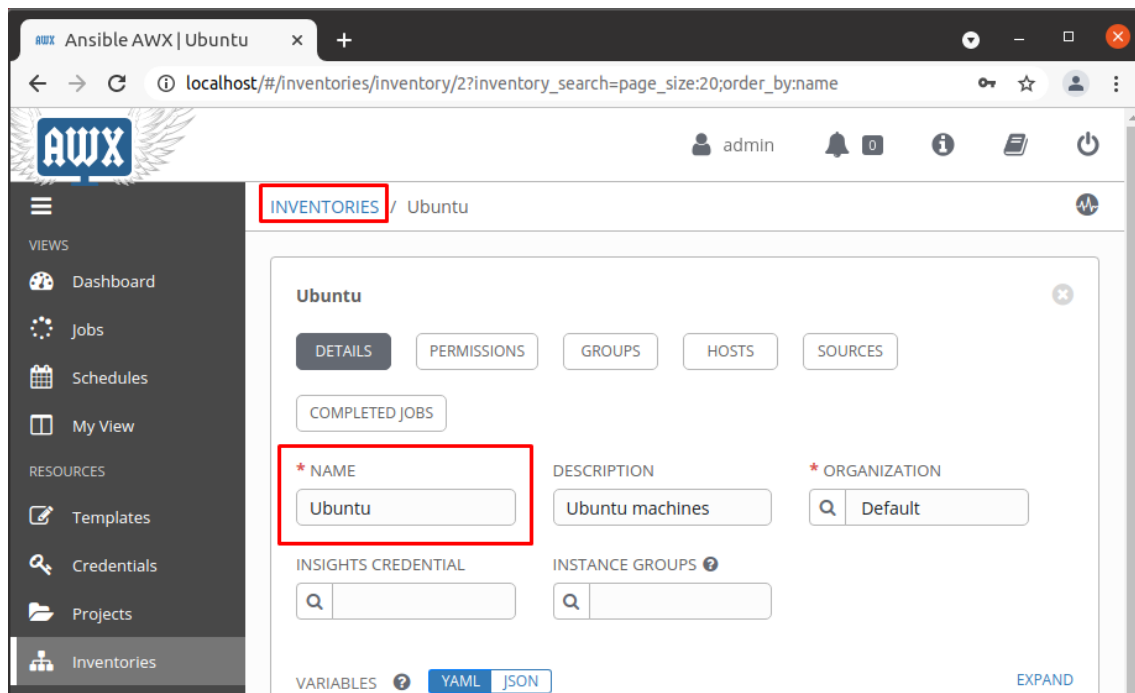
- NAME:** Ubuntu 18.04 (2)
- DESCRIPTION:** Credential machine Ubu
- ORGANIZATION:** Default
- CREDENTIAL TYPE:** Machine
- TYPE DETAILS:**
 - USERNAME:** alvaro
 - PASSWORD:** (field with 'ENCRYPT' button and 'Prompt on launch' checkbox)

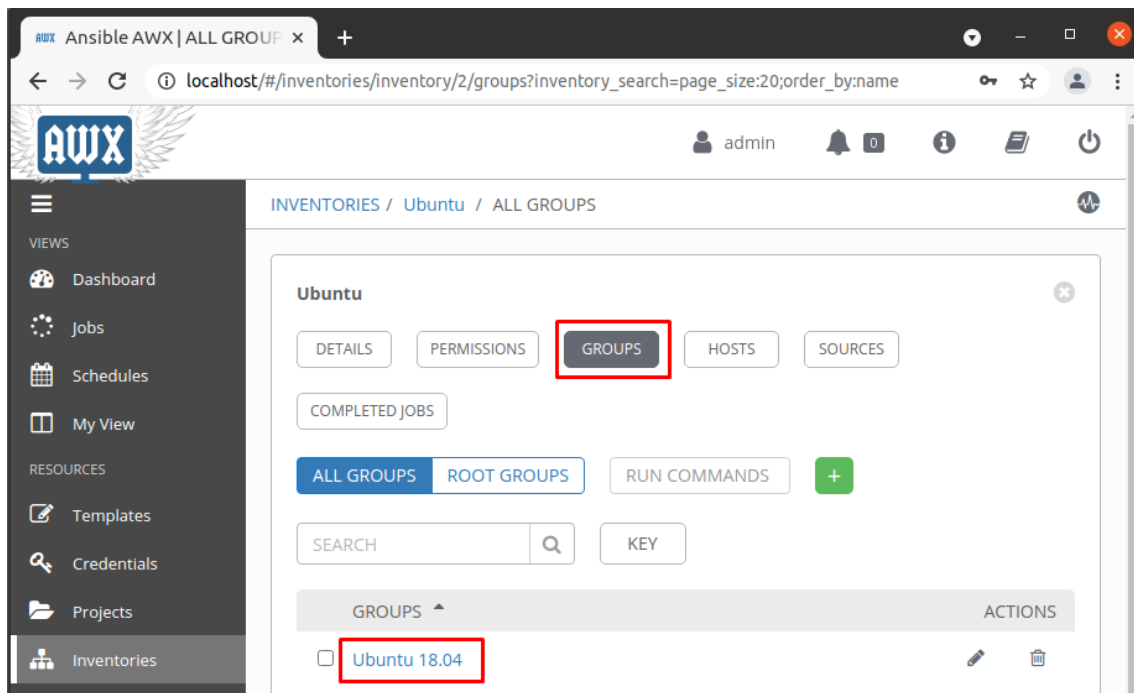
Para añadir los equipos nos vamos al apartado de "Inventories". En los Inventarios se guardan los equipos sobre los que queremos actuar.

Un inventario es una colección de hosts contra los que se pueden lanzar trabajos. Los inventarios se dividen en grupos y estos grupos contienen los hosts.

Creamos un inventario que se llame Ubuntu, en el vamos a crear un grupo llamado Ubuntu 18.04 y dentro metemos los hosts.

Para añadir los hosts podemos hacerlo mediante su dirección IP o con el hostname.





En mi caso he introducido la dirección IP de las máquinas. Para saber la dirección de un host introducimos el siguiente comando en el equipo:

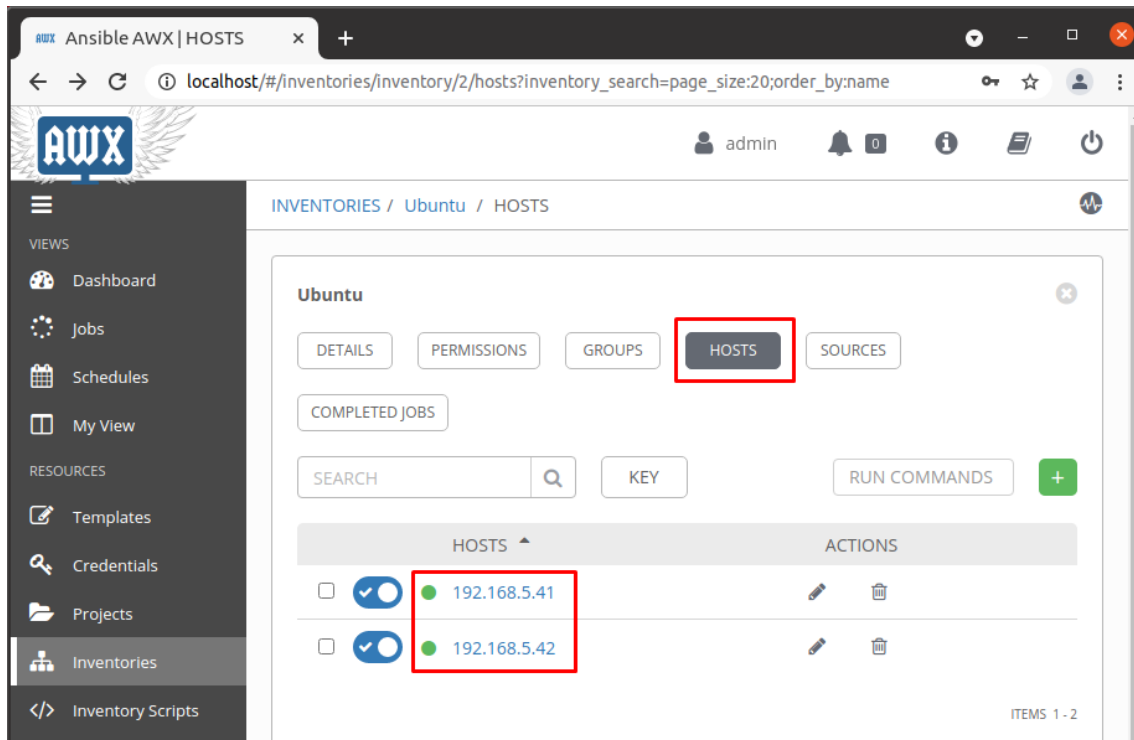
```
alvaro@alvaro-1:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.5.41 netmask 255.255.255.0 broadcast 192.168.5.255
    inet6 fe80::e427:e8f7:cbd2:e6c1 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:1f:08:90 txqueuelen 1000 (Ethernet)
    RX packets 24548 bytes 5351880 (5.3 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 476 bytes 58779 (58.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 163 bytes 14736 (14.7 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 163 bytes 14736 (14.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

alvaro@alvaro-1:~$
```

Para que el comando `ifconfig` funcione debemos tener instalado el paquete que se llama "net-tools". Si no lo queremos instalar podemos usar el comando "ip a" que viene instalado en Ubuntu por defecto.

```
sudo apt install net-tools -y
```



Cuando aparece el círculo de color verde significa que ha encontrado el host correctamente.

7.3. Ejecución de los scripts

7.3.2. ¿Cómo ejecutar los scripts en los equipos?

Nosotros no le mandamos el script directamente al equipo y él lo ejecuta, sino que tenemos que mandarle un archivo .yml que es el que el equipo interpreta y lleva a cabo.

Para ejecutar los scripts tenemos que crear una plantilla. Para ello nos vamos al apartado de “templates” y creamos una nueva:

The screenshot shows the Ansible AWX web interface. The browser address bar indicates the URL: `localhost/#/templates/job_template/12?template_search=page_size:20;order_by:name;type:wor...`. The page title is "TEMPLATES / Search update ubuntu". The left sidebar contains navigation links: VIEWS (Dashboard, Jobs, Schedules, My View), RESOURCES (Templates, Credentials, Projects, Inventories, Inventory Scripts), ACCESS (Organizations, Users, Teams), and ADMINISTRATION (Credential Types, Notifications, Management Jobs). The main content area is titled "Search update ubuntu" and includes tabs for DETAILS, PERMISSIONS, NOTIFICATIONS, COMPLETED JOBS, and SCHEDULES. Below the tabs is an "ADD SURVEY" button. The configuration form includes the following fields:

- * NAME:** Search update ubuntu
- DESCRIPTION:** (empty text box)
- * JOB TYPE:** Run (dropdown menu, with a "PROMPT ON LAUNCH" checkbox)
- * INVENTORY:** Ubuntu (dropdown menu, with a "PROMPT ON LAUNCH" checkbox)
- * PROJECT:** GitHub Project (dropdown menu)
- * PLAYBOOK:** ansible-security-patc... (dropdown menu)
- CREDENTIALS:** Ubuntu 18.... (dropdown menu, with a "PROMPT ON LAUNCH" checkbox)
- FORKS:** 0 (spin box)
- LIMIT:** (empty text box, with a "PROMPT ON LAUNCH" checkbox)
- * VERBOSITY:** 0 (Normal) (dropdown menu, with a "PROMPT ON LAUNCH" checkbox)
- JOB TAGS:** (empty text box, with a "PROMPT ON LAUNCH" checkbox)
- SKIP TAGS:** (empty text box, with a "PROMPT ON LAUNCH" checkbox)
- LABELS:** (empty text box)
- INSTANCE GROUPS:** (empty text box)
- JOB SLICING:** 1 (spin box)
- TIMEOUT:** 0 (spin box)
- SHOW CHANGES:** (toggle switch, with a "PROMPT ON LAUNCH" checkbox)
- OPTIONS:**
 - ☒ ENABLE PRIVILEGE ESCALATION

Tenemos que rellenar los siguientes datos de la plantilla:

- **Name:** El nombre que le queramos asignar a la plantilla. En este caso (Search update ubuntu)
- **Job type:** Podemos elegir entre dos opciones.
 - **Run:** Ejecutar el playbook y que nos muestre todo. Es la opción que elijo.
 - **Check:** Simula que lo ha ejecutado y nos mostraría como sería el resultado.
- **Inventory:** Sobre que inventario queremos actuar, en este caso el que hemos creado antes que contiene las 2 máquinas "Ubuntu".
- **Project:** Elegimos nuestro GitHub asociado previamente.
- **Playbook:** Aquí tenemos que elegir el archivo .yaml que es el que llama a nuestro script. Es el siguiente:

Está escrito en un lenguaje yaml. Es muy sencillo de programar y de comprender.

YAML es un formato para guardar objetos de datos con estructura de árbol. Sus siglas significan YAML (Ain't Markup Language).

Este lenguaje es muy legible para las personas, más legible que un JSON y sobre todo que XML.

Para que Ansible lo entienda le tenemos que dar la siguiente estructura:

- Name: Nombre que le queremos dar a la tarea a ejecutar.
- Hosts: sobre que equipos va a actuar.
- Taks:
 - name: nombre de la tarea.

A partir de aquí existe una lista de módulos que le podemos dar a los archivos que le mandamos ejecutar a Ansible. Todos están explicados en la [documentación](#)³.

La primera tarea "[task](#)"⁴ le pregunta a los host del AWX que distribución tienen instalada, en el primer caso es Ubuntu, esa misma tarea llama al siguiente playbook que es "[list_ubuntu.yaml](#)"⁵. En caso de que el host sobre el que actúa fuese Debian llamaría al archivo "[list_debian.yaml](#)"⁶.

³ Ver apartado REFERENCIAS o clicar en "documentación".

⁴ Ver apartado REFERENCIAS o clicar en "task" para ver el código.

⁵ Ver apartado REFERENCIAS o clicar en "list_ubuntu.yaml" para ver el código.

⁶ Ver apartado REFERENCIAS o clicar en "list_debian.yaml" para ver el código.

```
1 #First file
2 ---
3 - name: Find the Linux distribution
4   hosts: all
5   tasks:
6     - name: Classify hosts depending on their OS distribution
7       group_by:
8         key: "{{ ansible_distribution }}_{{ ansible_distribution_major_version }}"
9
10  - hosts: Ubuntu_18
11    gather_facts: False
12    tasks:
13      - name: Update for Ubuntu
14        include_tasks: list_ubuntu.yml
15
16  - hosts: Debian_9
17    gather_facts: False
18    tasks:
19      - name: Update for Debian
20        include_tasks: list_debian.yml
```

Este es el segundo archivo al que llama el primer playbook. Aquí le estamos diciendo que copie del GitHub el archivo que se llama "ubuntu_array.sh" y lo lleve a la ruta "/tmp/update.sh". Al ser un script tenemos que darle permisos de ejecución, le damos 0760 para que solo lo pueda ejecutar el usuario.

Y por último lo mandamos ejecutar con `bash /tmp/ubuntu_copy.sh`

```
1 #list_ubuntu.yml
2 ---
3 - name: Copy update.sh to server
4   copy:
5     src: ubuntu.sh
6     dest: /tmp/ubuntu_copy.sh
7     owner: alvaro
8     group: sudo
9     mode: '0760'
10  - name: Get a list of security updates
11    command: bash /tmp/ubuntu_copy.sh
12    ignore_errors: yes#
```

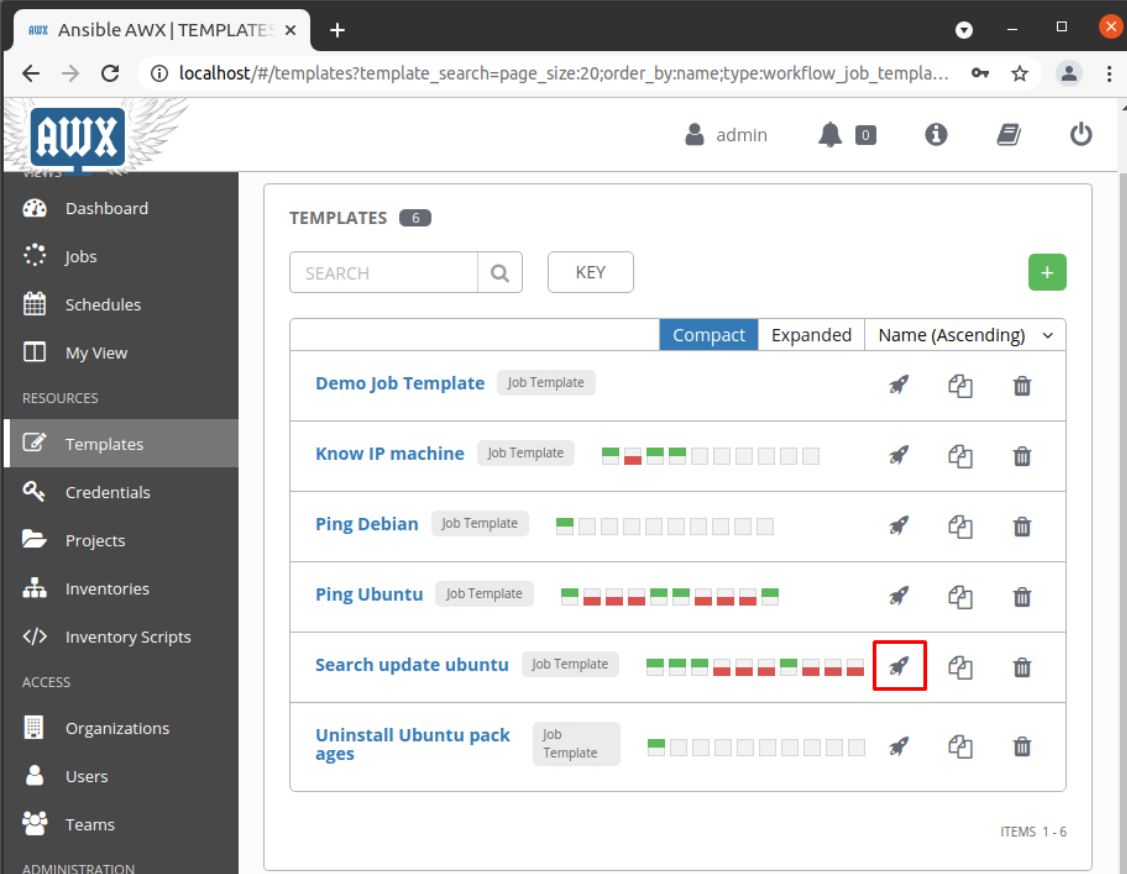
Visto graficamente que es como lo ejecutamos se haría de la siguiente manera.

Esa ha sido la explicación de lo que ocurre de fondo cuando lanzamos la plantilla.

Para lanzar la plantilla y que todo ocurra lo hacemos de la siguiente manera:

7.3.3. Lanzando los scripts

En la pestaña de Templates vemos todas las que tenemos creadas, le damos al cohete y se nos abre la plantilla donde vamos viendo lo que ocurre sobre el host.



The screenshot shows the Ansible AWX web interface. The left sidebar contains navigation links: Dashboard, Jobs, Schedules, My View, Resources, Templates (selected), Credentials, Projects, Inventories, Inventory Scripts, Access, Organizations, Users, Teams, and Administration. The main content area is titled 'TEMPLATES 6' and includes a search bar and a 'KEY' button. Below this is a table of templates with columns for Name, Job Template, and a status bar. The 'Search update ubuntu' template is highlighted with a red box around its rocket icon, indicating it is the one to be launched.

Name	Job Template	Status	Actions
Demo Job Template	Job Template	[Progress Bar]	[Launch] [Copy] [Delete]
Know IP machine	Job Template	[Progress Bar]	[Launch] [Copy] [Delete]
Ping Debian	Job Template	[Progress Bar]	[Launch] [Copy] [Delete]
Ping Ubuntu	Job Template	[Progress Bar]	[Launch] [Copy] [Delete]
Search update ubuntu	Job Template	[Progress Bar]	[Launch] [Copy] [Delete]
Uninstall Ubuntu pack ages	Job Template	[Progress Bar]	[Launch] [Copy] [Delete]

7.4. Resultado final del script⁷

Como resultado del script veríamos en formato .json los datos que sacamos de los diferentes equipos por ejemplo Ubuntu 1:

```

1  {
2      "msg": {
3          "repositories": [
4              {
5                  "repository": "http://us.archive.ubuntu.com/ubuntu",
6                  "packages": [
7                      {
8                          "name": "rsync",
9                          "release_date": "2020-02-18",
10                         "security": {
11                             "urgency": "medium",
12                             "cves": [
13                                 "CVE-2016-9840",
14                                 "CVE-2016-9841",
15                                 "CVE-2016-9842",
16                                 "CVE-2016-9843"
17                             ]
18                         }
19                     },
20                     {
21                         "name": "wget",
22                         "release_date": "2019-04-08",
23                         "security": {
24                             "urgency": "medium",
25                             "cves": [
26                                 "CVE-2018-20483",
27                                 "CVE-2019-5953"
28                             ]
29                         }
30                     }
31                 ],
32                 "metadata": {
33                     "status": "OK",
34                     "reason_status": "There are packages with security updates"
35                 }
36             }
37         ],
38         "metadata": {
39             "machine_hostname": "alvaro-1",
40             "generation_data_date": "2021-05-06 00:52:54"
41         }
42     },
43     "_ansible_verbose_always": true,
44     "_ansible_no_log": false,
45     "changed": false
46 }

```

En caso de que no haya actualizaciones de seguridad como es el caso del segundo host "Ubuntu 2" veríamos el siguiente resultado:

⁷ Ver apartado REFERENCIAS o clicar en el título "Resultado final del script" para ver el código

```

1  {
2      "msg": {
3          "repositories": [
4              {
5                  "repository": "",
6                  "packages": [],
7                  "metadata": {
8                      "status": "No packages to update",
9                      "reason_status": "There are no packages with security updates"
10                 }
11             }
12         ],
13         "metadata": {
14             "machine-hostname": "alvaro-2",
15             "generation_data_date": "2021-05-06 13:04:53"
16         }
17     },
18     "_ansible_verbose_always": true,
19     "_ansible_no_log": false,
20     "changed": false
21 }

```

7.4.2. Actualizar los paquetes listados:⁸

Una vez tenemos listados los paquetes que se pueden actualizar, si decidimos que sí que queremos llevar las actualizaciones a cabo tenemos que ejecutar la plantilla de actualización.

Consta del siguiente playbook:

Al igual que el primero lo que hace es identificar el sistema operativo sobre el que queremos actuar y lanza el siguiente playbook que es “ubuntu.yml”.

```

1  ---
2  - name: talk to all hosts for ansible distribution and ansible distribution major
3    hosts: all
4    tasks:
5      - name: Classify hosts depending on their OS distribution
6        group_by:
7          key: "{{ ansible_distribution }}_{{ ansible_distribution_major_version }}"
8
9  - name: Delete .json files before execute the playbooks
10   hosts: all
11   tasks:
12     - name: Deletes .json files
13       file:
14         path: /tmp/check_pending_security_packages.json
15         state: absent
16
17  - hosts: Ubuntu_18
18    gather_facts: False
19    tasks:
20      - name: Update for Ubuntu
21        include_tasks: ubuntu.yml
22
23  - hosts: Debian_9
24    gather_facts: False
25    tasks:
26      - name: Update for Debian
27        include_tasks: debian.yml

```

⁸ Ver apartado REFERENCIAS o clicar en el título “Actualizar los paquetes listados” para ver el código.

Este segundo playbook coge el script que sí que actualiza los paquetes y lo copia en /tmp/upgrade.sh. La segunda tarea es mandar ejecutarlo, el script de [actualización](#)⁹ es este.

```
1  ---
2  - name: Step 0. Copy upgrade_apt.sh to server
3    copy:
4      src: upgrade_os.sh
5      dest: /tmp/upgrade.sh
6      owner: alvaro
7      group: sudo
8      mode: '0766'
9
10 - name: Step 1. Get the list of security packages
11   command: bash /tmp/upgrade.sh list ubuntu
12   ignore_errors: yes
13   register: packages
14
15 - name: Step 2a. Installing new security patches...
16   apt:
17     name: "{{ packages.stdout.split() }}"
18     state: latest
19     register: aptout
20     when: packages.rc == 0
21
22 - debug: var=aptout
23
24 - name: Step 3a. Generate .json output
25   command: bash /tmp/upgrade.sh update "{{ packages.rc }}" ubuntu "{{ packages }}"
26   ignore_errors: yes
27   register: out
28   when: packages.rc == 0
29
30 - name: Step 2b. Getting execution trace when there is an error or no package to update
31   apt:
32     name: "{{ packages.stdout_lines }}"
33     state: latest
34     register: aptout
35     ignore_errors: yes
36     failed_when: not aptout.failed
37     when: (packages.rc == 1) or (packages.rc == -1)
38
39 - debug: var=aptout
40
41 - name: Step 3b. Generate .json output
42   command: bash /tmp/upgrade.sh update "{{ packages.rc }}" "{{ packages }}" ubuntu
43   ignore_errors: yes
44   register: out
45   when: (packages.rc == 1) or (packages.rc == -1)
```

Lanzado desde el servidor AWX tenemos que crear una plantilla como con el primer playbook.

Para ejecutarlo lo hacemos de la misma manera dándole al cohete.

⁹ Ver el apartado REFERENCIAS o hacer clic en “actualización”

The screenshot shows the Ansible AWX web interface. The browser address bar indicates the URL is localhost/#/templates/job_template/21?template_search=page_size:20;order_by:name;type:wor... The page title is 'Ansible AWX | Install updates ubuntu'. The sidebar on the left contains navigation links for VIEWS (Dashboard, Jobs, Schedules, My View), RESOURCES (Templates, Credentials, Projects, Inventories, Inventory Scripts), ACCESS (Organizations, Users, Teams), and ADMINISTRATION (Credential Types, Notifications, Management Jobs). The main content area is titled 'TEMPLATES / Install updates ubuntu'. It features a tabbed interface with 'DETAILS', 'PERMISSIONS', 'NOTIFICATIONS', 'COMPLETED JOBS', and 'SCHEDULES'. The 'DETAILS' tab is active, showing a form for configuring the job template. The form includes fields for NAME (Install updates ubuntu), DESCRIPTION, JOB TYPE (Run), INVENTORY (Ubuntu), PROJECT (GitHub Project), PLAYBOOK (ansible-security-patc...), CREDENTIALS (Ubuntu 18...), FORKS (0), LIMIT, VERBOSITY (1 (Verbose)), JOB TAGS, SKIP TAGS, LABELS, INSTANCE GROUPS, JOB SLICING (1), TIMEOUT (0), SHOW CHANGES (toggle), and OPTIONS (ENABLE PRIVILEGE ESCALATION checked).

Como resultado obtenemos el siguiente código:

Nos indica que paquetes se han actualizado, que son todos los listados en el primer script.

```

1  {
2    "msg": {
3      "repositories": [
4        {
5          "repository": "repository",
6          "metadata": {
7            "status": "OK",
8            "reason_status": "Packages updated",
9            "execution_trace": "{ 'changed': True, 'end': '2021-05-06 13:32:59.127372',
10          }
11        }
12      ],
13      "patch_date": "",
14      "metadata": {
15        "machine-hostname": "alvaro-1",
16        "generation_data_date": "2021-05-06 13:33:55"
17      }
18    },
19    "_ansible_verbose_always": true,
20    "_ansible_no_log": false,
21    "changed": false
22  }

```

8. CONCLUSIONES Y RECOMENDACIONES

Los conocimientos aprendidos durante el primer curso de ASIR en la asignatura de "Implantación de sistemas operativos" han sido muy útiles, ya que he trabajado mayoritariamente con Linux y en dicha asignatura se trabajan muchos conceptos que he podido poner en práctica en este trabajo.

La asignatura de "Administración de sistemas operativos" también ha sido de gran ayuda para este proyecto, ya que hemos trabajado con Servidores de Windows. El servidor AWX tiene cierta similitud a la hora de crear contraseñas, añadir equipos al servidor principal, etc.

Por parte de la empresa he trabajado con un servidor AWX que tienen montado y en funcionamiento, me han enseñado a lanzar las plantillas, a añadir los equipos al servidor, resolución de problemas típicos, etc.

Con la empresa también he trabajado el uso de Docker, levantando microservicios y desplegándolos en servidores más grandes como un ECS.

8.2. Proyección a futuro:

Ahora mismo lo que tenemos es un script que actualiza todos los paquetes de seguridad disponibles, como proyección a futuro se puede hacer un script interactivo que nos mostrase cada paquete de seguridad y nos preguntase si quiere actualizarse o no.

9. REFERENCIAS

Referencia 1 página 10: <https://paste.rs/fLQ.bash>

Referencia 2 página 21: <https://docs.ansible.com/ansible-tower/latest/html/userguide/credentials.html>

Referencia 3 página 29: <https://docs.ansible.com/ansible/latest/collections/ansible/builtin/index.html#plugins-in-ansible-builtin>

Referencia 4 página 29: <https://paste.rs/YLe.bash>

Referencia 5 página 29: <https://paste.rs/Lre.bash>

Referencia 6 página 29: <https://paste.rs/dy6.bash>

Referencia 7 página 32: <https://paste.rs/VWt.json>

Referencia 8 página 34: <https://paste.rs/62p.bash>

10. BIBLIOGRAFÍA

Páginas web

Ansible YT "Comenzando con Ansible"

<https://www.youtube.com/watch?v=qen8hmBWm1E>

Comandos GIT "dasdo/GIT.md"

<https://gist.github.com/dasdo/9ff71c5c0efa037441b6>

Course Linux academy "SkC O3 Engineering Students"

<https://linuxacademy.com/cp/learningpaths/enterprise/assigned/82114>

Curso OpenWebinars "¿Qué es Ansible?"

<https://openwebinars.net/academia/recurso/video/5247/>

Josphat Mutai "How To Install Ansible AWX on Ubuntu 20.04 | 18.04 Linux"

<https://computingforgeeks.com/how-to-install-ansible-awx-on-ubuntu-linux/#ex1>

Lukman LAB "How to Install Ansible AWX on Ubuntu Server 20.04 LTS"

<https://www.youtube.com/watch?v=lcx8FR4d6Pk>

Miquel Mariano Ramis "Ansible AWX - Part 1 – Instalación"

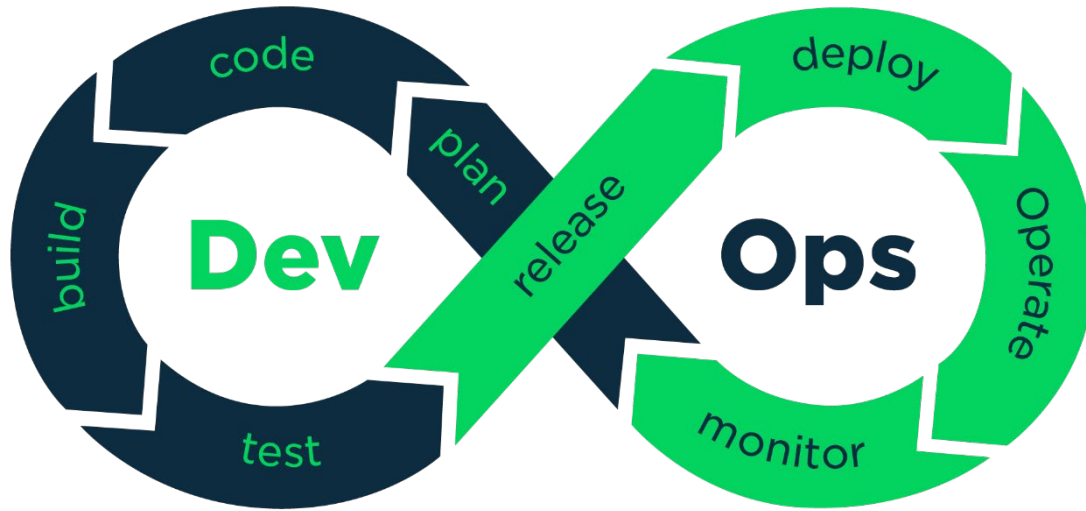
<https://miquelmariano.github.io/2019/01/23/ansible-awx-installation/>

Libros

Geerling, Jeff. *Ansible for DevOps: Server and configuration management for humans*. 2015. ISBN: 098639341X

Geerling, Jeff. *Ansible for Kubernetes: Automate app deployment on any scale with Ansible and K8s*. 2018.

11. ANEXOS



ANSIBLE CHEAT SHEET

Ansible

- It is an open source engine that automates deployment, orchestration, cloud provisioning and other tools.
- It uses a playbook to describe jobs and uses YAML, which is human readable.
- It is designed for multi-tier deployment. It is agentless and works by connecting nodes through ssh.

How Does it Work?

- Connects nodes and pushes small programs called modules to them and are removed when they are done.
- The management node controls whole execution of the playbook.
- The inventory file provides the list of hosts where the modules need to be run.
- The management node does an 'ssh' connection and executes the modules and installs the software.

Troubleshooting

- Common strategies to debug playbooks are:
 - Debug and register
 - Use verbosity (verbosity level)
- Playbook issues:
 - Quoting
 - Indentation
- Some drawbacks are:
 - OS restrictions: is OS dependent so code on one OS will not work for another
 - Once playbook is running, adding of hosts is not possible
 - Error reporting is mediocre.

Environment Setup

Types of machines:

- Control machine:** manages other machines
- Remote machines:** controlled by other machines

Multiple remote systems can be handled by one machine.

- Remote machine managing is done by ansible by default.
- Ansible doesn't leave any software running on them. Therefore there is no need of an upgrade when moving to a newer version.
- Install it through apt, yum, pip, OpenCSW
- Installing it through apt:


```
$ sudo apt-get update
$ sudo apt-get install software-properties-common
$ sudo apt-add-repository ppa:ansible/ansible
$ sudo apt-get update
$ sudo apt-get install ansible
```
- Run ansible version to make sure it was installed properly.

YAML

- YAML syntax is used to express ansible playbooks
- Key-value pair:** Dictionary is represented in key value pair. Ex:


```
name: James John
rol: 34
div: 8
sex: male
```
- Representing lists:**
 - Each element has to be written in a new line with "-" as the prefix.
 - countries:
 - America
 - Iceland
- Lists inside the dictionary:**

```
name: James John
rol: 34
div: 8
sex: male
likes:
  - english
```
- Boolean terms are also used in YAML.

Advantages of Ansible

- It is free and open source.
- Agentless. No master client model.
- System requirements.
- Developed in python.
- Lightweight and quick deployment.
- Ansible uses YAML syntax in config files.
- Large community base.

Ad-hoc Commands

- General syntax of ad-hoc command:


```
Command hostgroup module/options[arguments]
```

FUNCTION	COMMANDS
Check connectivity of hosts	<code>#ansible -c group -m ping</code>
Rebooting hosts	<code>#ansible -c group -a "/bin/reboot"</code>
Check host system's info	<code>#ansible -c group -m setup less</code>
Transferring files	<code>#ansible -c group -m copy -a "src=home/ansible dest=/tmp/home"</code>
Create new user	<code>#ansible -c group -m user -a "name=ansible password='encrypted passwords'"</code>
Deleting user	<code>#ansible -c group -m user -a "name=ansible state=absent"</code>
Check if package is installed and update it	<code>#ansible -c group -m yum -a "name=httpd state=latest"</code>
Check if package is installed and don't update it	<code>#ansible -c group -m yum -a "name=httpd state=present"</code>
Check if package is a specific version	<code>#ansible -c group -m yum -a "name=httpd 1.8 state=latest"</code>
Check if package is not installed	<code>#ansible -c group -m yum -a "name=httpd state=absent"</code>
Starting a service	<code>#ansible -c group -m service -a "name=httpd state=started"</code>
Stopping a service	<code>#ansible -c group -m service -a "name=httpd state=stopped"</code>
Restarting a service	<code>#ansible -c group -m service -a "name=httpd state=restarted"</code>

Playbooks

- It is the place where all YAML files are stored and executed. Acts like a to-do list.
- YAML, yet another markup language
- A playbook can have more than one plays. Plays map the instructions defined against a particular host
- Typically written in a text editor like notepad or notepad++

Sample playbook/YAML file:

```
name: Install and configure DB
hosts: testServer
become: yes
vars:
  oracle_db_port_value: 1521
tasks:
  - name: Install the Oracle DB
    yum: <code to install the DB>
  - name: Ensure the installed service is enabled
    service:
      name: <your service name>
```

- Tags of YAML:**
 - Name:** name of the playbook
 - Hosts:** specifies the list of hosts. Tasks can be on the same machine or a different one.
 - Vars:** defines the variables which you can use
 - Tasks:** It is the list of action that needs to be performed. A task is always linked to a module.

Variables

- Same as using variables in programming languages. Ex:


```
hosts: <your hosts>
tomcat_port: 8080
Here tomcat_port is assigned to Bobo
```
- Keywords used:**
 - Block:** ansible syntax to execute a block
 - Name:** name of the block
 - Action:** the code that is to be executed
 - Register:** registers the output
 - Always:** states that below word will be run
 - Msg:** displays the message
- Exception handling:**
 - Similar to any other programming language
 - Keywords: rescue and always
 - The code is written in block
 - It goes to the rescue phase and gets executed if the command in the block fails.
 - Thereby block is the same as "try block", catch block is like "rescue" and always performs the same function as we know.

Terms

- Service/Server:** a process that provides service
- Machine:** physical machine, Vm or a container
- Target machine:** end machine to be configured by ansible
- Task:** an action
- Playbook:** location where YAML files are written and executed

YAML

- Indentado con espacios
 - Miembros de las listas -
 - Uno por línea
 - [] varios separados por ,
 - Vector asociativo Clave: valor
 - Uno por línea
 - { } varios
 - --- inicio de documento
 - ... fin de documento
 - Nodos marcados con & se pueden repetir con *
 - # comentario de línea
-
- Listas (iguales, distinta representacion)
 -
 - A
 - B
 - C
 -
 - [A, B, C]
 - Vector asociativo (iguales, distinta representacion)
 -
 - Nombre: Pepe
 - Edad: 30
 -
 - {nombre: Pepe, edad: 30}
-
- Para el formato correcto en bloque cada clave (campo) debe ir denajo de la otra.
 - EJEMPLO:
 - Correcto:
 -
 - name: Nombre del host
 - hosts: host01
 - become: yes
 - task:
 - name: muestra hola por pantalla
 - command: echo "Hola"
 - Incorrecto:
 -
 - name: Nombre del host
 - hosts: host01
 - become: yes
 - task:
 - name: muestra hola por pantalla
 - command: echo "Hola"