ANSIBLE

Permite ejecutar comandos sobre máquinas remotas. Instalamos Ansible en un equipo, conectamos ese equipo por ssh a otros equipos, y desde el primero podemos enviarles órdenes.

Ejemplo de uso: si tenemos muchos equipos conectados y queremos apagarlos todos, con Ansible no es necesario ir apagándolos de uno en uno, sino que podemos apagar todos del tirón.

Ansible es un lenguaje declarativo (decimos el estado final que queremos que tenga la máquina). Funciona con dos cosas: Ad-Hocs (comandos individuales, los que veremos en prácticas) y Playbooks (como scripts de shell).

Guía de instalación (viene en la documentación de Ansible). El profe recomienda instalarlo en el anfitrión ya que en el examen vamos a necesitar 3 máquinas: anfitrión y dos MVs conectadas.

Ansible solo se instala en la máquina que lo ejecuta, en el resto solo hace falta que tengan SSH+Python (lo tenemos que configurar en las MVs). RECOMENDACIÓN: copiar las llaves públicas para no tener que estar poniendo la contraseña.

Empezamos la práctica.

Arrancamos las máquinas y para empezar a ejecutar Ansible debemos conectarnos por ssh el anfitrión con las MVs.

Primero trabajamos con la MV de Ubuntu. En el anfitrión comprobamos que nos podemos conectar por SSH:
ssh erodriguez@IP_Ubuntu
Ctrl^D

Configuración en: /etc/ansible/ansible.cfg

INVENTARIO: lista de servidores con los que vamos a trabajar. Lo vemos en:
more /etc/ansible/hosts → en la parte de [webservers] vemos los servidores agrupados. De esta manera, si ejecutamos una orden sobre [webservers], se ejecutará en cada uno de los servidores del conjunto. Es muy útil.

Además, podemos ponerles *NOMBRES* a los servidores. Escribimos en el inventario:
sudo nano -w /etc/ansible/hosts
> correo1 ansible_host=192.168.56.105 → Al equipo de Ubuntu lo llamo correo1. Esta línea la podemos escribir donde sea, yo la he puesto al final de 'Ungrouped hosts'.

Módulo 'ping':

Cuando mandamos un comando por Ansible, lo que hacemos es mandar un archivo de Python via SSH.

PING de ansible: comprueba que puede conectarse a la máquina por SSH y que puede ejecutar un programa en Python (no es como el ping que conocemos). Podemos obtener información en los modulos de la documentación de ansible.
ansible correo1 -m ping -u erodriguez : Se hace el ping con éxito!

Indicamos en el inventario el nombre de usuario de cada máquina (ya que puede ser distinto en cada una). Añadimos a lo que habíamos escrito antes:
> correo1 ansible_host=192.168.56.105 **ansible_user=erodriguez**

Si ahora hacemos ping, no hace falta especificar el usuario:
ansible correo1 -m ping : Se hace el ping con éxito!

Otro ejemplo. Aquí podemos mandar un mensaje con el ping:
ansible correo1 -m ping -a “data=’Hola practicas ise’”

Configuramos ahora la otra máquina (Rocky) que llamaremos ‘lamp’. En el inventario añadimos:
sudo nano -w /etc/ansible/hosts
> lamp ansible_host=192.168.56.106 ansible_user=erodriguez ansible_port=<puerto de ssh> : El puerto es necesario ponerlo si hemos cambiado el puerto de SSH y no es el de por defecto (22).
ansible all -m ping : envía un ping a todas las máquinas (en este caso las dos que tenemos).

También, podemos ponerles *ETIQUETAS* a los servidores y cada equipo puede estar en varias etiquetas (por ejemplo: servidor de correo, servidor de lamp…).
Ahora en el inventario modificamos y añadimos debajo de [webservers]:
[mysql]
lamp
Con esto le hemos puesto una etiqueta al servidor lamp. Si tuviéramos más equipos, podríamos añadirlos debajo y se “agruparían” en una misma etiqueta.
ansible mysql -m ping : sólo se lo manda a la máquina de Rocky (la de lamp).

Módulo ‘shell’:

Nos permite ejecutar comandos de Shell en máquinas remotas.

ansible mysql -m shell -a “ps ax” : ver todos los procesos que están en ejecución en el servidor de lamp.
ansible mysql -m shell -a “who” : ver los usuarios que están conectados al sistema y también otras informaciones como cuándo se arrancó el sistema y cuál es el nivel de ejecución del sistema.
ansible mysql -m shell -a “who | grep -i David” : ver información sobre el usuario David.

Comando ‘poweroff’:

Solo lo puede ejecutar el root pero pide contraseña y no queremos pasarla !! Para ello podemos configurar que un usuario privilegiado se pueda convertir en superusuario sin necesidad de poner la contraseña.

En Rocky hacemos:

sudo visudo : vamos al final donde habla del grupo ‘wheel’. Descomentamos la línea de NO_PASSWD:ALL y comentamos la línea de ALL (la de encima).
sudo bash : ya no pide contraseña

En Ubuntu:
sudo visudo : vamos al grupo %sudo y cambiamos ALL (el último) por NOPASSWD:ALL.
sudo bash : ya funciona, podemos entrar sin contraseña.

En el anfitrión:
ansible webservers -a “poweroff” : apaga todos los usuarios. Esto da error porque cuando ejecutamos un comando de usuario privilegiado hay que pasarle un parámetro.
ansible webservers -a “poweroff” --become : para que se “convierta” en superusuario. Se apagan las máquinas.

USO DE CPU

Directorio /proc/:

Monta el SO (el kernel de Linux), no se corresponde a los sistemas de ficheros reales con existencias permanentes. Se crea en memoria cada vez que arranca el kernel. Los números son los ID de los procesos en ejecución. Se puede hacer cd a uno de esos números.

cd /proc/929/ : carpeta del proceso 929
ls
sudo more mem : para dar información sobre el funcionamiento interno (memoria del proceso).
cd .. : nos situamos en /proc.
more cpuinfo : información útil sobre la CPU.

MUY IMPORTANTE:
more loadavg : nos da las medias del uso de CPU del último min, de los últimos 5 min y de los últimos 10/15 min (realmente nos da el número medio de tareas que han estado ejecutándose o esperando a ser ejecutadas). También nos indica el número de procesos en ejecución frente al número de procesos totales, y el id del proceso del último proceso que ha entrado en ejecución. Ejemplo:

- 1 CORE:
  Si sale 0.11 en la primera columna significa que en el último min había de media ¿11 tareas? que estaban en ejecución o esperando a ser ejecutadas.
  Si sale 0.5 significa que se ha estado ejecutando el 50% de la CPU. Si sale 1 significa que en el último min ha habido 1 sólo proceso ejecutándose todo el rato (CPU al 100%).
  Si sale 2 significa que la CPU está al 200%, y si tenemos un único core, esto implicaría que un proceso se ha estado ejecutando todo el rato y otro ha estado esperando a ser ejecutado. Que sea mayor que 1 es crítico.

- 4 CORES:
  Hay que dividir el número por 4 y nos da el porcentaje de CPU.
  Por ejemplo si ha salido 1, en realidad se ha estado usando en total la CPU un 25%.

Todo esto también lo podemos ver con los comandos top y uptime.

Comando 'stress':

Desde el anfitrión:
sudo apt install stress : nos permite crear procesos para que saturen la CPU enviando muchos procesos (de E/S, de acceso a disco, etc).
watch -n1 cat /proc/loadavg : cada segundo nos muestra la carga media de la CPU. Lo dejamos abierto.

Desde otra MV nos conectamos por ssh al anfitrión:
stress --cpu 8 --io 4 --vm 2 : ejecutar 8 procesos que saturen la CPU, 4 procesos girando en sincronización sync() saturando lam E/S, y otros 2 procesos saturando memoria escribiendo y leyendo continuamente de su espacio de memoria virtual (por defecto 256 Mb).

Comando ‘top’:

Con watch estamos viendo en el anfitrión la carga media de CPU que se va actualizando. Si queremos ver toda la información de los procesos que se están ejecutando, usamos:
top

Tenemos que entender bien toda la información que nos muestra este comando:

- Tasks: Nos dice el número de tareas, cuántas en ejecución, durmiendo, parados y zombie (parados y zombie lo normal es que están a 0). Zombie = está muerto pero está pendiente de liberar recursos.
- %CPU: nos dice el porcentaje de CPU según procesos de usuario, del sistema, nice (procesos priorizados por encima de lo que les corresponde), idle (que no están haciendo nada), waiting (esprando entrada salida), interrupción hardware, interrupción software.
- MiB Mem: MUY IMPORTANTE. Uso de memoria. Nos dice los megas totales, que desglosa en libres, utilizados, asignados a buffer y caché (lecturas, escrituras, etc) y memoria disponible (mide la memoria máxima que un proceso puede demandar como máximo).
- MiB Swap: Indicadores de memoria virtual, no es bueno que haya uso de memoria virtual, lo ideal es que estuviese a 0. Si un servidor está continuamente usando memoria virtual preocuparos, estáis tirando cpu.
- Procesos del sistema

Avail mem (swap) > *Free mem (Mem)* : porque la memoria libre es aquella que no se está usando para nada (desperdiciada), mientras que la memoria disponible es la que se puede asignar a nuevos procesos o a los ya existentes.

Es preferible que el uso de memoria virtual sea 0. Si no, estaríamos desperdiciando CPU.

Comando ‘free’:

Desde la MV: Paramos el stress con ^C.
free : Nos da información sobre la memoria parecida a la del comando top pero aparece también ‘shared’ (memoria compartida).

Tanto free como top sacan la información de los siguientes archivos:
cd /proc
more meminfo
more vmstat

Comando ‘uptime’:

uptime : nos da info sobre la carga media de CPU. Se corresponde con lo que aparece en el fichero /proc/loadavg.
more uptime : solo tiene dos números: por cuántos segundos el sistema ha estado encendido, y cuantos segundos ha estado inactivo.

CRON

Es un servicio que podemos gestionar con systemctl. Lo que hace es leer un archivo de configuración que le dice lo que tiene que ejecutar y cuándo hacerlo. Se usa para programar tareas recurrentes normalmente de mantenimiento (backups, limpieza…) en Linux.

Archivo de configuración: /etc/crontab
cd /etc/
more crontab : nos dice bajo qué shell se ejecutan los comandos, el path que va a usar (es mejor usar paths absolutos para que los encuentre bien. El inconveniente aparece cuando movemos un archivo de un directorio a otro y hay que actualizar el path).
En el archivo, *run-parts* hace que se ejecuten todos los archivos ejecutables de la carpeta indicada.
También podemos indicar bajo qué usuario se van a ejecutar los archivos en la columna 'user-name'.

Ejemplo:
17 * * * * erodriguez comando : ejecuta el comando desde mi usuario en el minuto 17 de todas las horas de todos los días de todos los meses del año, todos los días de la semana.

Comando logger:

Los logs o registros del sistema son ficheros de texto que registran cronológicamente la totalidad de actividades e incidencias importantes que ocurren en el sistema operativo o red. Linux registra absolutamente todo lo que pasa en el sistema operativo mediante los logs.

El comando logger introduce un mensaje en log del sistema. Ver man logger.

which logger : nos dice dónde esta el fichero con los logs.
echo "Prueba logger" | logger : introduce el mensaje en el system log

Vamos a crear nuestro proprio archivo cron en nuestro espacio de usuario (no privilegiado):

crontab -e → elegir 'nano' para editarlo. Es distinto del anterior porque no deja indicar el usuario (es lógico, te pone por defecto a ti como usuario).
Añadimos : ' * * * * * echo "Practicas Ise" | /usr/bin/logger -t "ISE" ' y guardamos.
tail -f /var/log/syslog : al final vemos los echos

ZABBIX

Es una herramienta de monitorización de propósito general que recaba información general de métricas del sistema y las almacena en una base de datos. Podemos visualizarla de forma gráfica con los datos a lo largo del tiempo.

Es muy vérsatil porque puede monitorizar servicios, programas de usuario…

Está el Zabbix Server y el Zabbix Agent: el Server le pide métricas a cada Agent (dime el loadavg, dime el n.º usuarios actualmente, dime si Apache está funcionando, dime los tiempos de no se qué, etc). Para ello hace PULL. Zabbix también admite PUSH (los agentes le envían info al servidor) pero es poco habitual. Por ejemplo si hay muchos Agents y no queremos que el servidor esté haciendo pull todo el rato.

El Server tiene una base de datos (nosotros usaremos SQL) donde estan definidos los hosts (máquinas q vamos a monitorizar). Tiene tablas internas donde guarda lo que recaba y con lo que se pueden generar gráficas. Presenta los datos en forma de página web. Por eso necesita LAMP (mysql, php y http). Los Agentes sólo tienen que darle acceso al Server a los puertos por donde le envían la info (hay que abrir el firewall, darle acceso, los puertos y alguna que otra cosa).

IMPORTANTE: Antes de empezar la instalación tenemos que instalar LAMP (descargamos MySQL en vez de MariaDB).

Instalamos Zabbix Server en la MV de Ubuntu desde su página web con la opción de ‘por paquetes’ y nos vamos a Debian Ubuntu. Instalamos la versión 5 (la versión 6 da problemas), en Ubuntu, y seleccionamos Server, MySQL y Apache.
Seguimos paso por paso lo que nos viene ahí.
!! En el paso c. al crear el usuario hay que cambiar ‘password’ por nuestra contraseña.
!! En el recuadro siguiente donde pone “-p zabbix”, cambiar ‘zabbix’ por nuestra contraseña.
En el penúltimo paso nos metemos en http://IP/zabbix y ahí tenemos que terminar la instalación. El puerto dejamos el que viene por defecto.

Por último, vamos a Quickstart del menú de la página web de Zabbix. En el login se pone como username Admin y como contraseña "zabbix" o la que hayamos puesto.

En el apartado de Configuración:
- *Hosts groups:* son agrupaciones de hosts según el criterio del administrador. Por ejemplo: todas las máquinas Linux, todas las máquinas LAMP. NO es lo mismo que las etiquetas de Ansible.
- *Templates*: Plantillas con métricas predefinidas, alarmas… para asignarlas a los hosts que queramos. Nosotros vamos a usar los que vienen ya definidos.
- *Items*: distintas métricas que hay para una cosa en concreta.
- *Applications*: agrupaciones de items (métricas). Una métrica puede tener asociado un *trigger*, esto es, una alarma que se activa cuando ocurre algo (esto no lo vamos a ver).

Vamos a crear un Host para monitorizar la máquina de Rocky. Nos metemos en Hosts y vemos que está el Server (podemos ver los templates, applications…).

Antes de crear el Host, tenemos que instalar el Zabbix Agent en la MV de Rocky. Lo hacemos de nuevo desde la página web de Zabbix.

Le damos a instalación, por paquetes, y elegimos: version 5 de Zabbix, Rocky, version 9 del OS, , y seguimos las intrucciones del paso 2 (hay que hacerlo como root, ejecutar: sudo bash).

Una vez instalado:
sudo systemctl status zabbix-agent : no está activo
sudo systemctl start zabbix-agent
sudo systemctl status zabbix-agent : ya sí.
sudo firewall-cmd --list-all : no está el puerto de zabbix.
sudo firewall-cmd –add-service=zabbix-agent : podríamos añadir el puerto pero así nos aseguramos de que lo estamos haciendo bien (nos podríamos equivocar de puerto).
firewall-cmd --list-all : ya sí aparece
firewall-cmd --runtime-to-permanent : salvamos los cambios

Tenemos que darle permiso a la máquina de Ubuntu para que consulte el Agente (por defecto solo permite consultar al localhost (en este caso, al Server):
cd /etc/zabbix
nano -w zabbix-agentd.d : A la línea “Server = 127…” le añadimos: “, 192.168..0/24” para que de acceso a todos los de la red.
systemctl restart zabbix-agent

Ahora comprobamos que ya nos podemos conectar con el Zabbix Agent enviándole una petición.

Comandos de Zabbix. Ejemplos:
system.cpu.load[,avg5] : devuelve el loadavg de los últimos 5 min.
proc.mem[,root] : memoria usada por todos los procesos ejecutándose como root.

Queremos enviar una petición (parecido a curl). En Ubuntu instalamos también el Zabbix-Agent (no se hace desde la página web.
sudo apt install zabbix-agent
sudo apt install zabbix-get
man zabbix-get : copiamos el comando de ejmplo y lo ejecutamos
zabbix_get -s 192.168.56.20 (IP de Rocky) -p 10050 (puerto de zabbix) -k “system.cpu.load[,avg5]”
: Si no da error es que la comunicación va bien.

Ahora ya podemos crear el host de Rocky:
En la pág web de Zabbix > Quickstart > Config > Hosts le damos a Crear Host.
Nombre: Rocky, Group: Linux Servers, IP la de la MV de Rocky, todo lo demás por defecto.
Añadimos templates:

- Template OS Linux by Zabbix agent (si da error lo quitamos porque puede ser que ya lo tenga)
- Template Module Linux CPU by Zabbix agent
- Template Module Linux memory by Zabbix agent
- Template App SSH Service
- Template App Apache by HTTP

Ahora vamos a pág web de Zabbix > Quickstart > Monitoring > Latest data.
Indicamos el host y ponemos el filtro: Application = Memory. Podemos ver los datos gráficamente.

EJERCICIO: Vamos a monitorizar SSH.

Ponemos ahora de filtro ‘Application = SSH Service’ y hacemos zoom en la gráfica (ultimos 15 min) para que se vea bien.

Vamos a apagar SSH y luego vamos a volver a encenderlo, y vamos a ver cómo cambian los datos.

1. Nos metemos en la MV de Rocky:
sudo systemctl stop sshd
systemctl status sshd

2. Vamos a la página web y en algún momento veremos que se actualiza.
TRUCO para que se actualicen los datos inmediatamente: Configuración > Hosts > Item > Items de SSH > Entramos y damos al botón Execute now. Volvemos a donde veíamos la gráfica y recargamos la página.
Vemos que la gráfica decrece drásticamente.

3. Nos metemos en la MV de Rocky
sudo systemctl start sshd
systemctl status sshd

4. Vamos a la página web y en algún momento veremos que se actualiza. Podemos usar el TRUCO de nuevo. La gráfica crece de nuevo.

En el Dashboard se pueden desactivar las alarmas (Acknowledge).

EXAMEN:

Ansible : Anfitrión con Ansible y dos MVs para monitorizar. Nos puede pedir ping, poweroff, comandos, inventario, tags…

Zabbix : MV Ubuntu con Zabbix Server (se instala automáticamente el Agente) y MV Rocky con Zabbix Agent instalado pero no configurado. SSH y LAMP también podemos llevarlo instalado y configurado.
Lo que nos puede pedir es dar de alta al host (firewall y todos los comandos para configurarlo). Tendremos que hacer caps de pantalla de la página de Zabbix de crear el host y todo eso.

Cron: nos puede pedir que lancemos un Ansible cada 5 min por ejemplo.