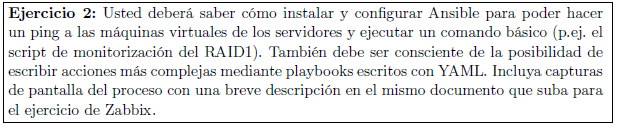
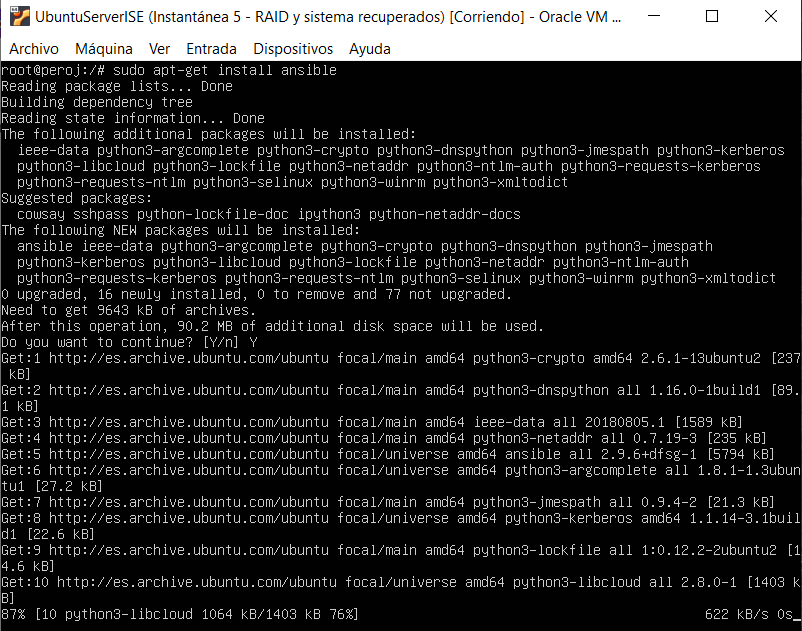
## MONITORIZACIÓN Y “PROFILING”

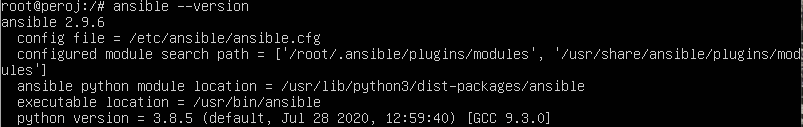
## ANSIBLE



1º) Instalamos ***ansible*** en UbuntuServer con ***sudo apt-get install ansible***. Comprobamos la versión con ***ansible --version***.



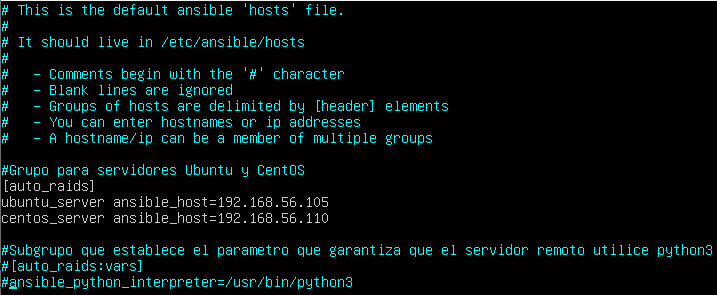
Y vemos que usa la versión 3.8.5 de Python, es decir, el módulo *python3* para ejecutar los scripts *.py*.



2º) Ahora deberemos irnos al archivo de inventario con ***vi /etc/ansible/hosts*** donde indicaremos los host con las IPs de los servidores con los que queremos tratar, en nuestro caso, tanto de Ubuntu como de CentOS (podemos ver varios ejemplos de configuración comentados en el archivo).

Este archivo se suele utilizar también para configurar variables que serán válidas sólo para hosts o grupos específicos, a fin de usarse dentro de los *playbooks* y las plantillas. Algunas variables también pueden afectar la forma en que se ejecuta un *playbook*, como la variable *ansible\_python\_interpreter* que veremos a continuación.

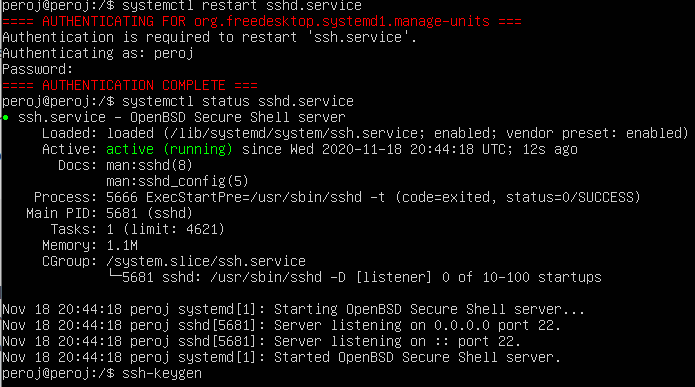
Creamos un grupo de servidores llamado *auto\_raids* donde incluiremos las IPs tanto de CentOS como de Ubuntu. También creamos (si es necesario) un subgrupo de este llamado *auto\_raids:vars* donde establecemos el parámetro ***ansible\_python\_interpreter=/usr/bin/python3*** (descomentar sólo si es necesario).



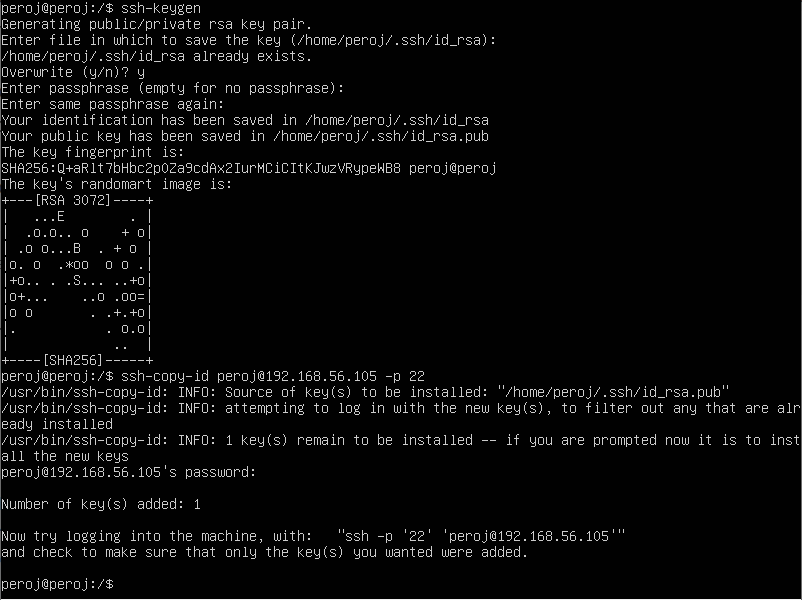
Comprobamos que se han añadido los host correctamente al inventario con ***ansible-inventory --list -y***.

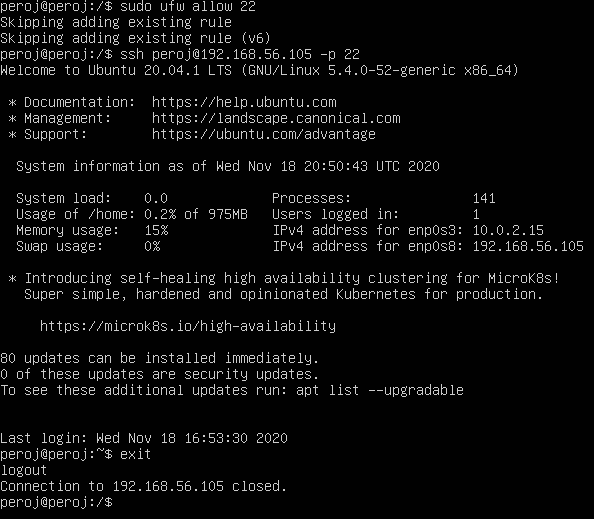


3º) Antes de probar conexión, debemos cambiar el puerto de las máquinas al puerto por defecto (22), con lo que esto conlleva, en Ubuntu deberemos simplemente avisar al firewall, pero en CentOS también deberemos actualizar el SELinux y ya después, volver a generar los pares de claves pública-privada, tanto de Ubuntu consigo mismo como con CentOS (ya que *ansible* se conecta por ssh).

-En Ubuntu: Abrimos ***/etc/ssh/sshd\_config*** y cambiamos el puerto al *Port 22* y el parámetro *Password Autentication yes* para poder generar las nuevas claves (después lo cambiamos otra vez)*.* Después de guardar el fichero, reiniciamos el servicio con ***systemctl restart sshd.service*** para aplicar los cambios y lo vemos con ***systemctl status sshd.service***. Nos aseguramos que el puerto 22 está abierto en el firewall ***ufw allow 22*** *(*que modifica las *iptables* para que nos permita comunicación por este puerto).

Primero generamos la clave en el cliente (Ubuntu) con ***ssh-keygen*** . Se la mandamos al servidor (Ubuntu) con ***ssh-copy-id*** [***peroj@192.168.56.105***](mailto:peroj@192.168.56.105) ***-p 22,*** metemos la contraseña de Ubuntu y vemos que se ha añadido una clave. Finalmente, comprobamos haciendo login por ***ssh*** [***peroj@192.168.56.105***](mailto:peroj@192.168.56.105) ***-p 22***, ya no nos pide la contraseña.

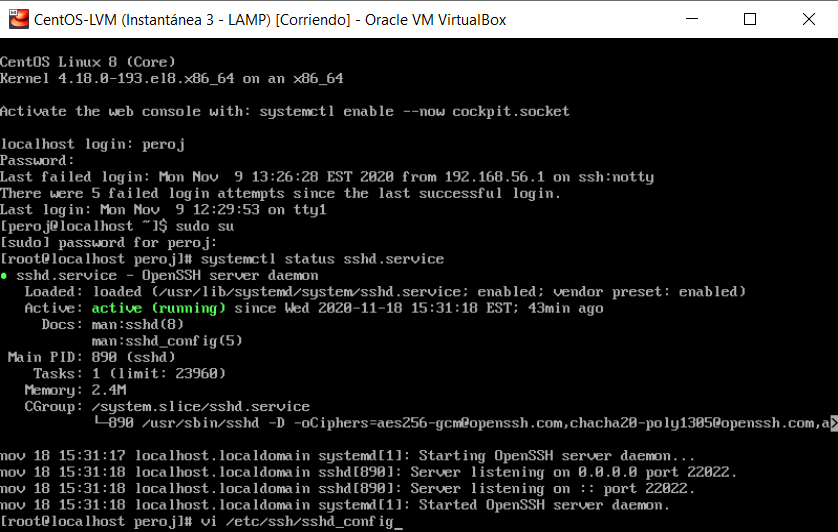




Ya tenemos conexión por ssh de Ubuntu consigo misma a través del puerto 22.

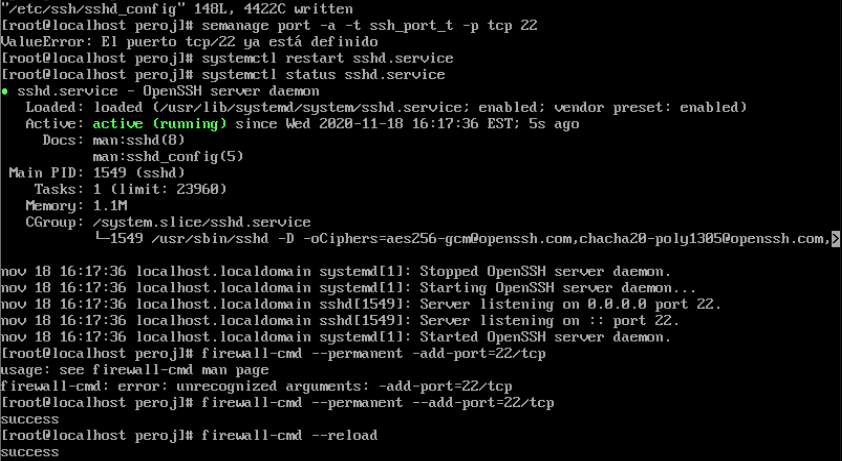
En Ubuntu debemos recordar poner el parámetro de ***/etc/ssh/sshd\_config*** *AutenticationPassword no*.

-En CentOS: Haremos “el mismo” proceso anterior para configurarle el ssh server y podernos conectar desde UbuntuServer. Comprobamos que esta dado de alta el servicio con ***sudo systemctl status sshd.service***. Vamos a cambiarle también al puerto 22 y el *PasswordAutentication* a *yes* modificando el fichero ***sudo vi /etc/ssh/sshd\_config.***



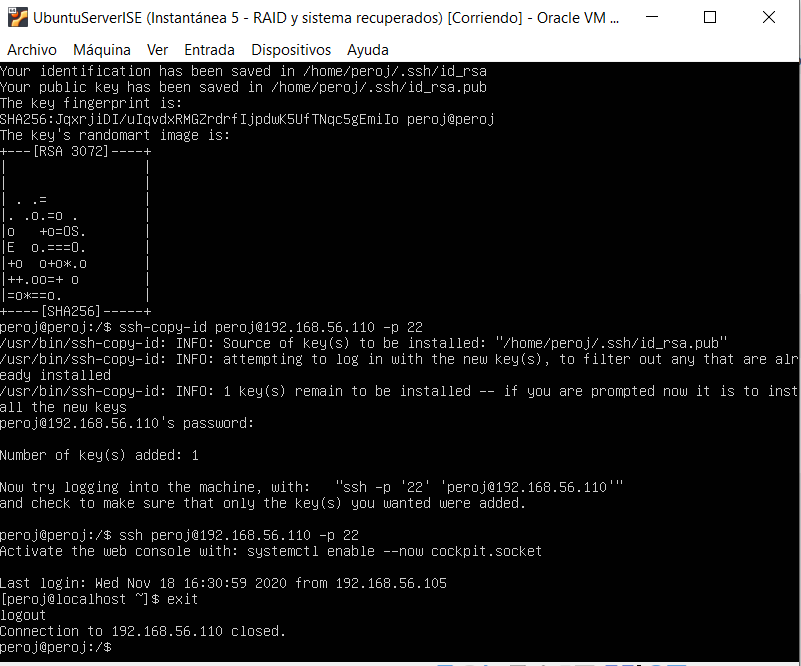
Una vez hecho esto, debemos avisar al sistema de seguridad de Linux SELinux de que hemos cambiado el puerto con la orden ***semanage port -a -t ssh\_port\_t -p tcp 22*** y reiniciamos el servicio con ***sudo systemctl restart sshd.service***.

También avisamos al cortafuegos de que permita conexiones por ese puerto con ***sudo firewall-cmd --permanent --add-port=22022/tcp*** y recargamos las reglas con ***sudo firewall-cmd --reload***.

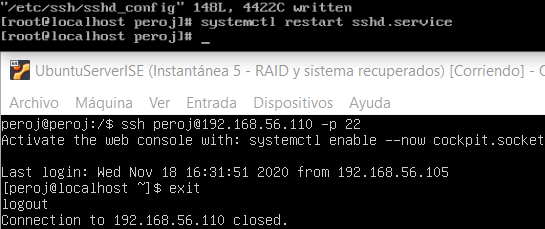


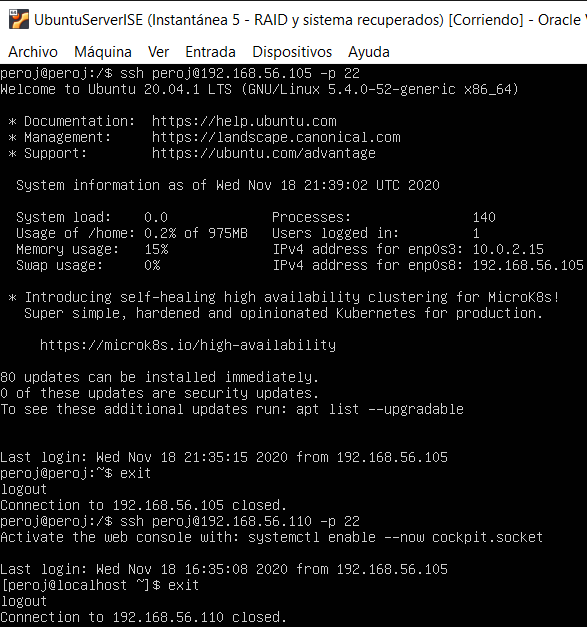
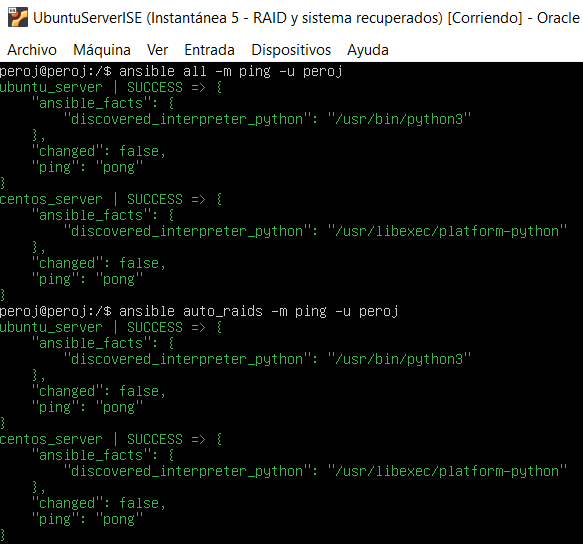
Ya si intentamos conectarnos desde Ubuntu a CentOS con ***ssh*** [***peroj@192.168.56.110***](mailto:peroj@192.168.56.110) ***-p 22*** ya podemos sin problema.

Ahora debemos enviar desde Ubuntu la clave pública-privada a CentOS para que no sea necesaria la contraseña.

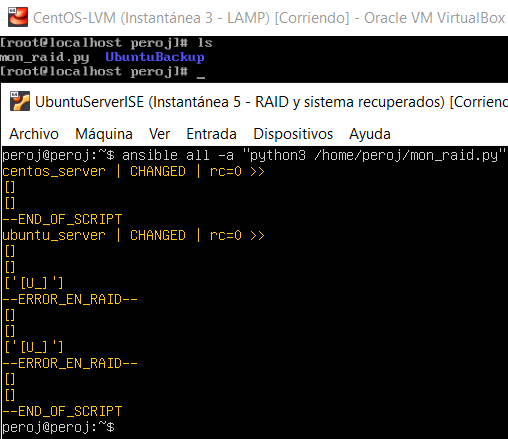


Y finalmente, le cambiamos del fichero de configuración de CentOS ***sudo vi /etc/ssh/sshd\_config*** el *PasswordAuthentication* a no*,* para que solo se pueda conectar un usuario que previamente esté vinculado con una clave pública-privada. Y reiniciamos el servicio con ***sudo systemctl restart sshd.service***.

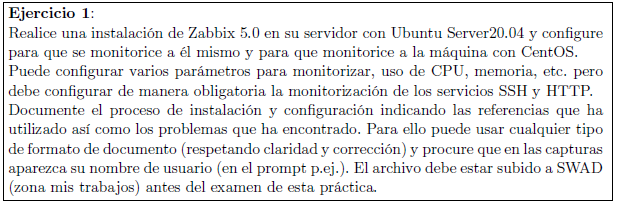


4º) Una vez que tenemos correctamente configuradas las conexiones ssh procedemos a comprobar que Ansible nos devuelve el ping correctamente a los servidores con ***ansible all -m ping -u peroj*** o ***ansible auto\_raids -m ping -u peroj.***

5º) Ahora debemos crear el script ***mon\_raid.py*** en CentOS (y en la misma ubicación que en Ubuntu). También podemos enviarlo por ssh desde Ubuntu a CentOS.

Ahora podemos ejecutarlo con Ansible en todos los servidores y recibir el resultado en nuestra máquina con ***ansible all -a “python3 /home/peroj/mon\_raid.py”*** (si el fichero a ejecutar no estuviera en la misma ubicación en las dos máquinas, esto deberíamos hacerlo con un script de configuración o *playbook*). Comprobamos y vemos como en Ubuntu nos devuelve el error en los RAID pero en CentOS no, dado que en CentOS (aunque es la versión sin RAID) funcionan correctamente los discos duros.

## ZABBIX



(toda la información obtenida durante la realización del ejercicio ha sido del manual oficial de Zabbix5.0)

1. ***Conceptos básicos Zabbix***

Es un software de monitorización que crea un *frontend* en el navegador que permite ver de forma visual los envíos de los agentes (la información que envían los servidores monitorizados).

Tenemos lo que se conoce como:

* Servidor --> Es el que tiene la recolección de datos. Procesa los datos para monitorizar lo que deseemos de los agentes. Vamos a tener uno, Ubuntu.
* Agentes --> Son los que le envían los datos al servidor. Vamos a tener dos agentes, CentOS y Ubuntu.

Una vez que instalemos y configuremos Zabbix tanto en CentOS como en Ubuntu, trabajaremos siempre desde el *frontend* en el servidor Ubuntu.

*ARQUITECTURA ZABBIX*

Zabbix está formado por varios componentes software:

* SERVER --> Zabbix Server es el componente central al que los agentes reportan o devuelven la información y estadísticas de disponibilidad e integridad. El servidor es el repositorio central en el que se almacenan todos los datos de configuración, estadísticos y operativos.
* DB STORAGE --> Toda la información de configuración, así como los datos recopilados por Zabbix, se almacenan en una base de datos.
* WEB INTERFACE --> Para un fácil acceso a Zabbix desde cualquier lugar y desde cualquier plataforma, se proporciona la interfaz web. La interfaz es parte del servidor Zabbix y generalmente (pero no necesariamente) se ejecuta en la misma máquina física que la que ejecuta el servidor.
* PROXY --> El proxy Zabbix puede recopilar datos de rendimiento y disponibilidad en nombre del servidor Zabbix. Un proxy es una parte opcional de la implementación de Zabbix; sin embargo, puede ser muy beneficioso distribuir la carga de un solo servidor Zabbix. (No vamos a usarlos aquí)
* AGENT --> Los agentes Zabbix se implementan en objetivos de monitoreo para monitorear activamente los recursos y aplicaciones locales e informar de los datos recopilados al servidor Zabbix. Desde Zabbix4.4, hay dos tipos de agentes disponibles:
  + Zabbix Agent: Ligero, compatible con muchas plataformas, escrito en C.
  + Zabbix Agent2: Extraflexible, facilmente ampliable con complementos, escrito en Go.

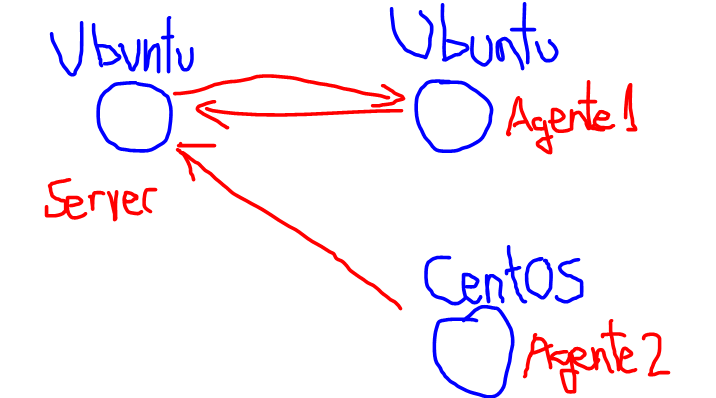
*FLUJO DE DATOS GENERAL*

Para crear un elemento que recopile datos, primero debe crear un *host*. Pasando al otro extremo del espectro de Zabbix, primero debemos tener un *item* (elemento) para crear un disparador. Debemos tener un disparador para crear una acción (*action*). Por lo tanto, si deseamos recibir, por ejemplo, una alerta de que la CPU está sobrecargada en el servidorX, primero debemos crear una entrada de host (*host entry*) para el servidorX seguida de un *item* para monitorear la CPU, luego un disparador (*trigger*) que se activa si la CPU es demasiado alta, seguido de un *action* que le envía un correo electrónico, por ejemplo.

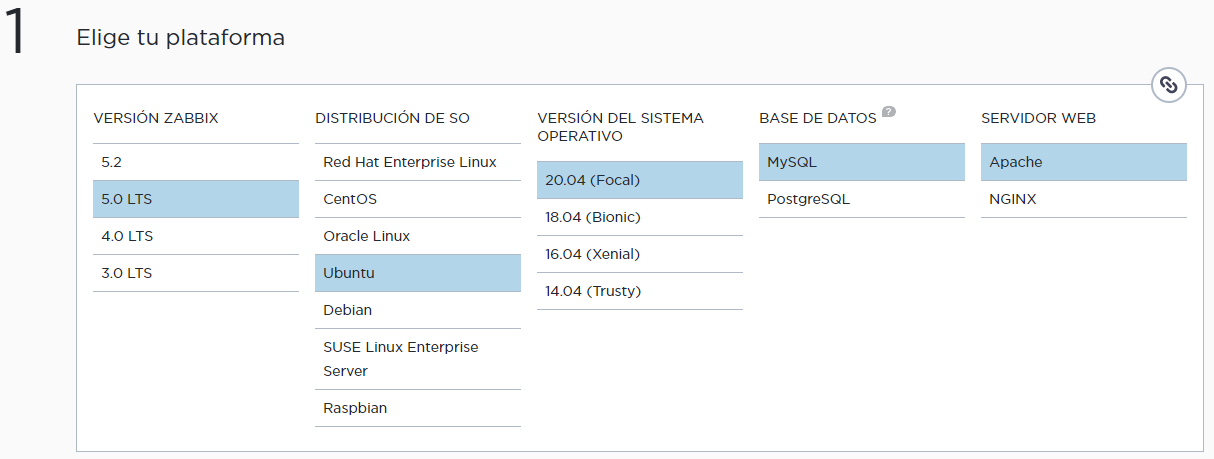
Aunque parezcan muchos pasos, con el uso de plantillas realmente no son tantos. Sin embargo, gracias a este diseño es posible crear configuraciones muy flexibles.

*INSTALACIÓN*

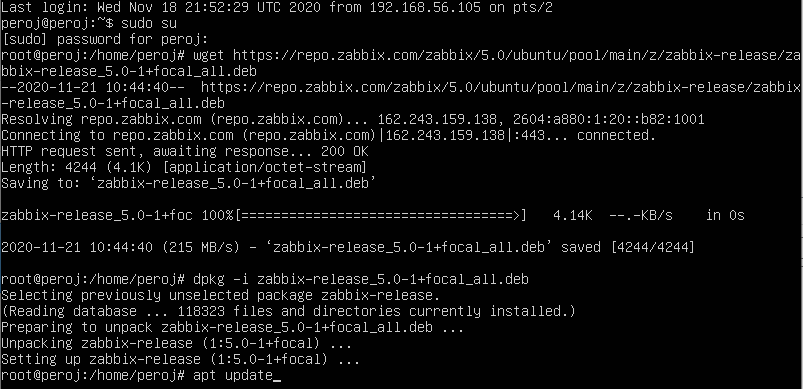
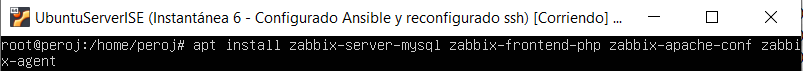
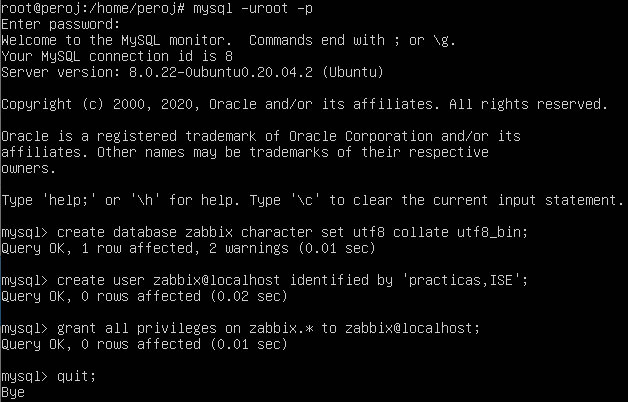
<https://www.zabbix.com/download>



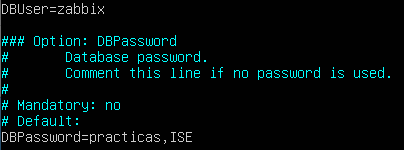
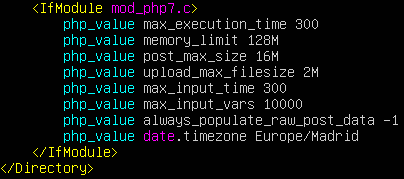
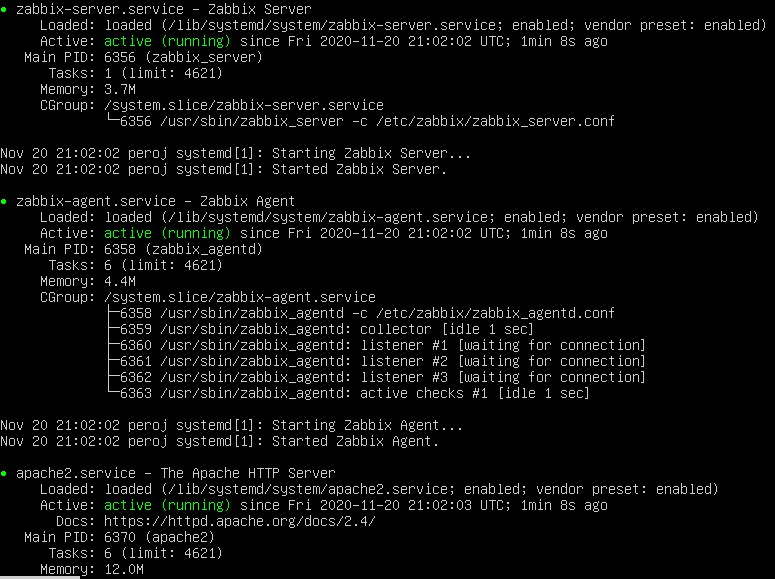
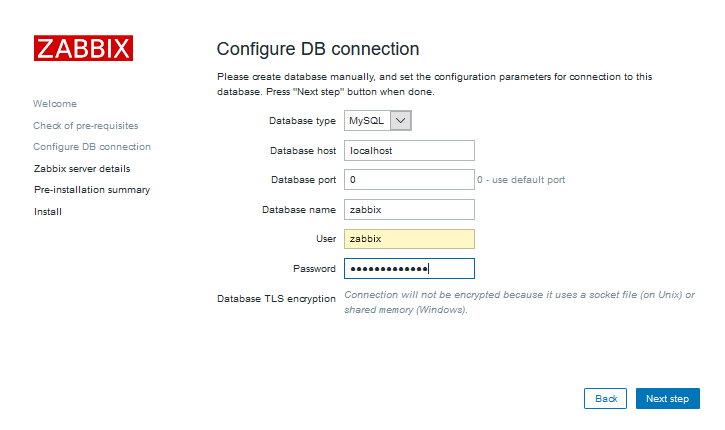
Por tanto, deberemos instalar tanto el servidor como el agente en la máquina con UbuntuServer. Y sólo el agente en CentOS.

1. ***Instalación y Configuración en Ubuntu - Zabbix Server y Zabbix Agent:***

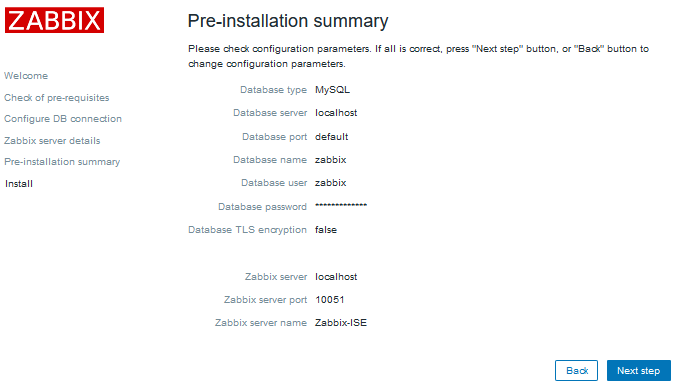
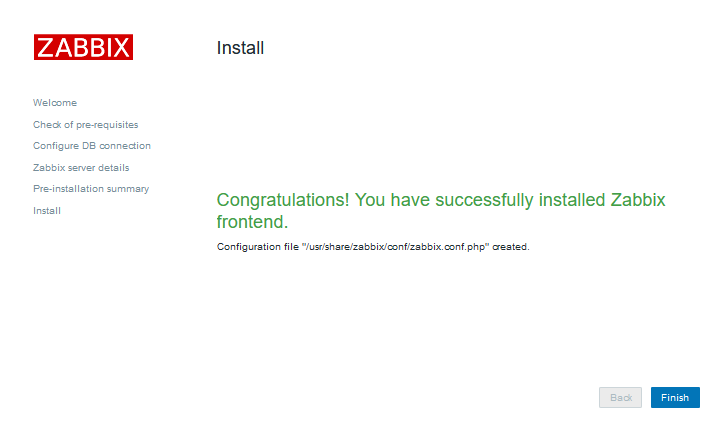
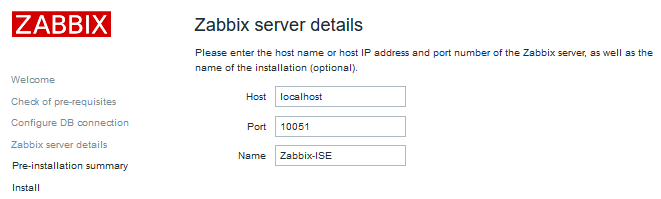
Y seguimos los pasos que se nos indica más abajo.

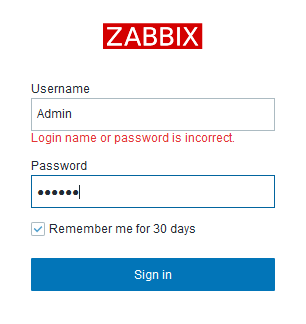
1. Instalar el repositorio de Zabbix:
2. Instalar el servidor, la interfaz y el agente de Zabbix:
3. Crear base de datos inicial en el host de base de datos:

En el servidor Zabbix, importamos el esquema inicial y los datos iniciales. Se pide que ingresemos contraseña recién creada (*practicas,ISE*).

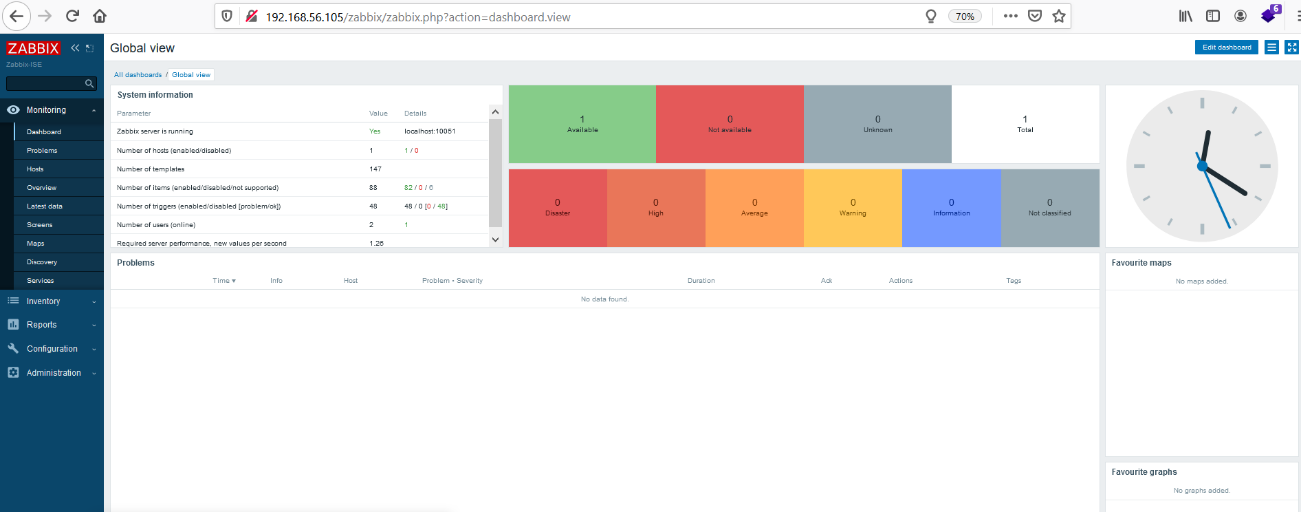
1. Configurar la base de datos para el servidor Zabbix. Editamos el archivo ***/etc/zabbix/zabbix\_server.conf***:
2. Configurar PHP para la interfaz de Zabbix. Editamos el archivo ***/etc/zabbix/apache.conf***, descomentamos y configuramos nuestra zona horaria:
3. Iniciamos los procesos del agente y del servidor Zabbix y los configuramos para que se inicien con el sistema:
4. Configurar la interfaz de Zabbix. Nos conectamos a la interfaz Zabbix recién instalada a http://192.168.56.105/zabbix:

Password: *practicas,ISE*

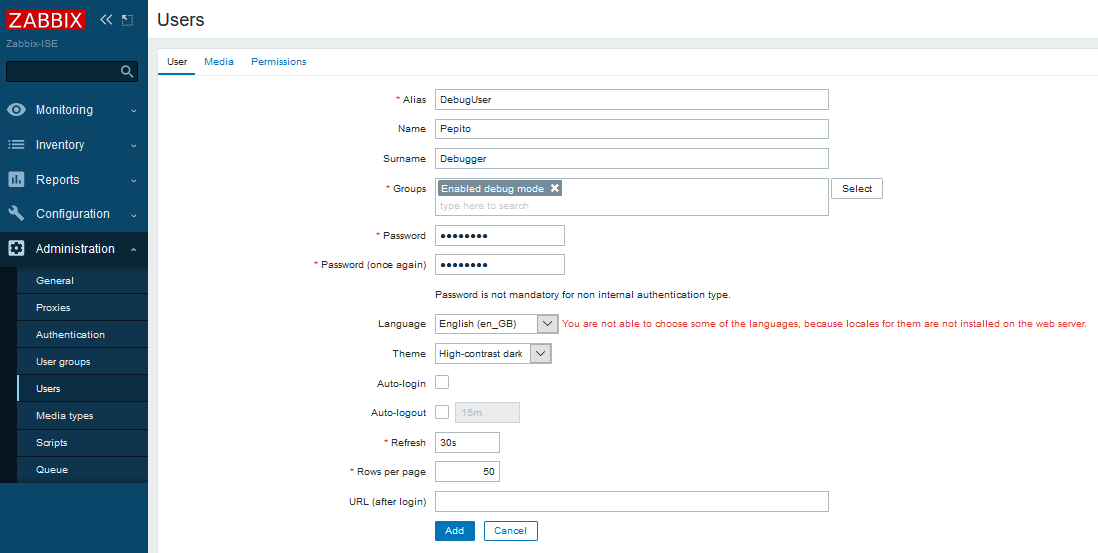


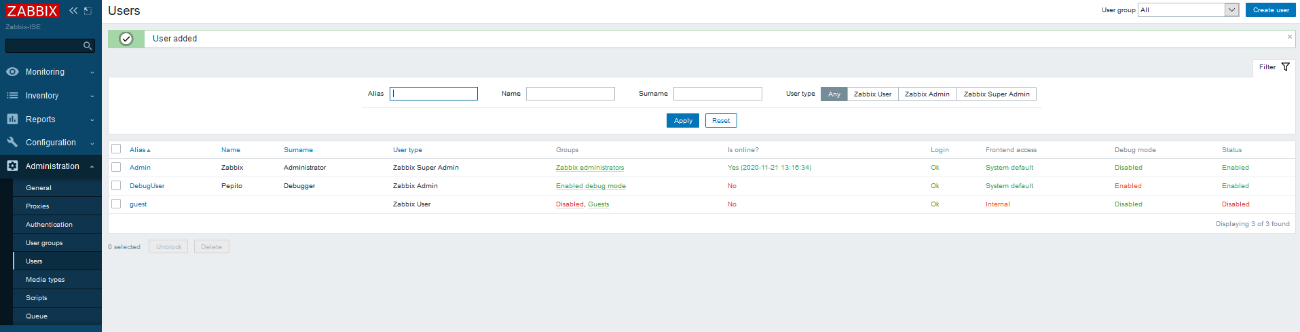
Username: Admin

Password: zabbix

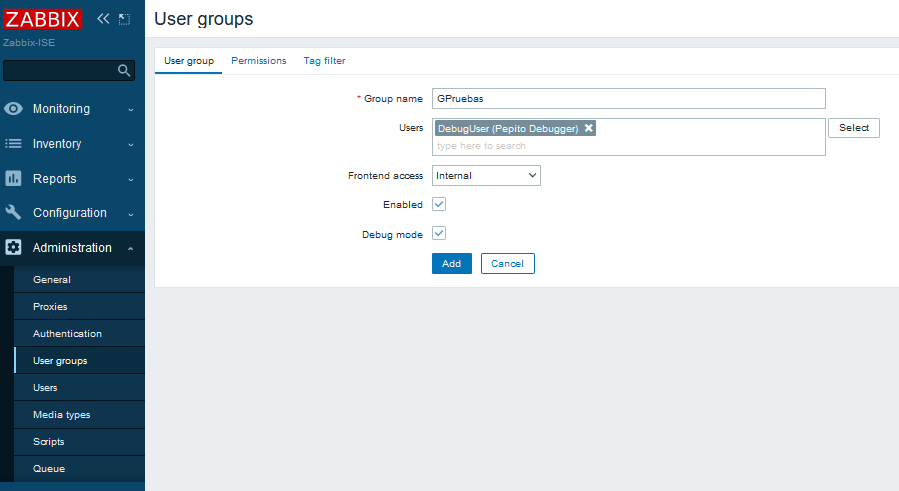
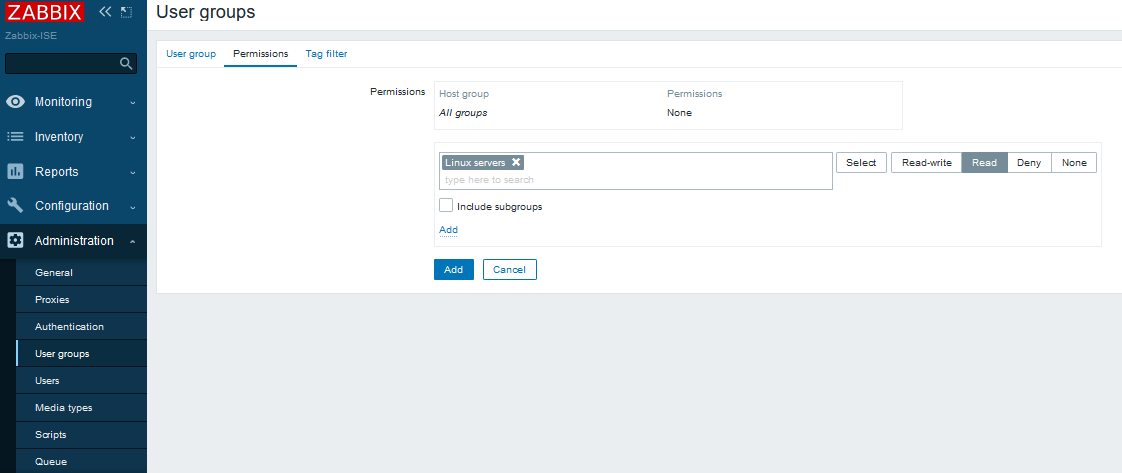
Si introducimos 5 veces mal los datos de acceso la interfaz de zabbix se pausará durante 30 minutos para prevenir ataques de fuerza bruta. Las IPs de los intentos fallidos se mostrarán al iniciar sesión correctamente.

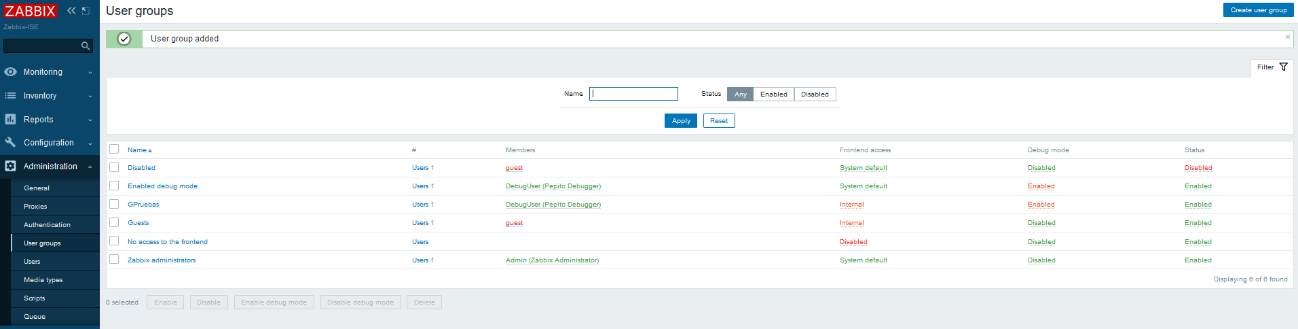
Probamos a añadir un usuario nuevo.

Fuente: <https://www.zabbix.com/documentation/5.0/manual/quickstart/login>

Contraseña para el nuevo usuario *pepito*: *debugger*

Por defecto, los usuarios nuevos no tienen permisos de acceso al host.

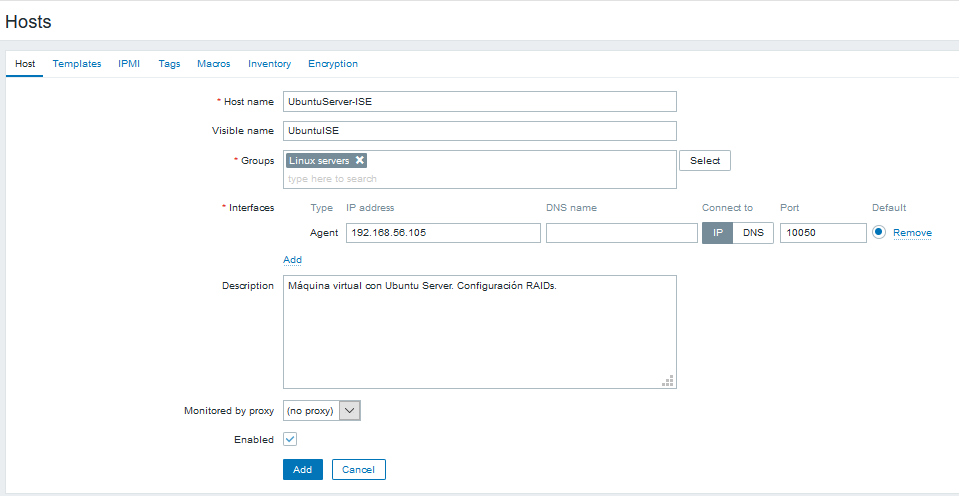
Entonces, debemos saber que los permisos solo se pueden asignar a grupos de usuarios no a usuarios individuales. Vamos a probar a crear un nuevo grupo de usuarios donde esté el usuario *pepito* y que adquiera acceso de solo lectura al grupo de servidores de Linux.

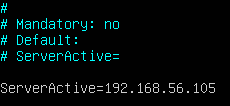


Para añadir nuevas máquinas a monitorear nos vamos a *Configuracion/Hosts/CreateHost* como se indica en el manual de Zabbix5.0: <https://www.zabbix.com/documentation/5.0/manual/quickstart/host>

Un host en Zabbix es una entidad en red (física, virtual) que desea monitorizar. La definición de lo que puede ser un “host” en Zabbix es bastante flexible ya que puede ser un servidor físico, un conmutador de red, una máquina virtual o alguna aplicación.

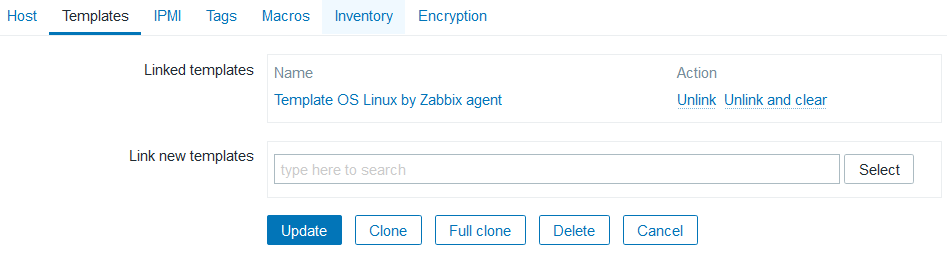
Por defecto vemos que ya tenemos creado el host *Zabbix server*. Pero debemos modificarle la dirección IP para indicar que el servidor Zabbix está en nuestra máquina Ubuntu Server.

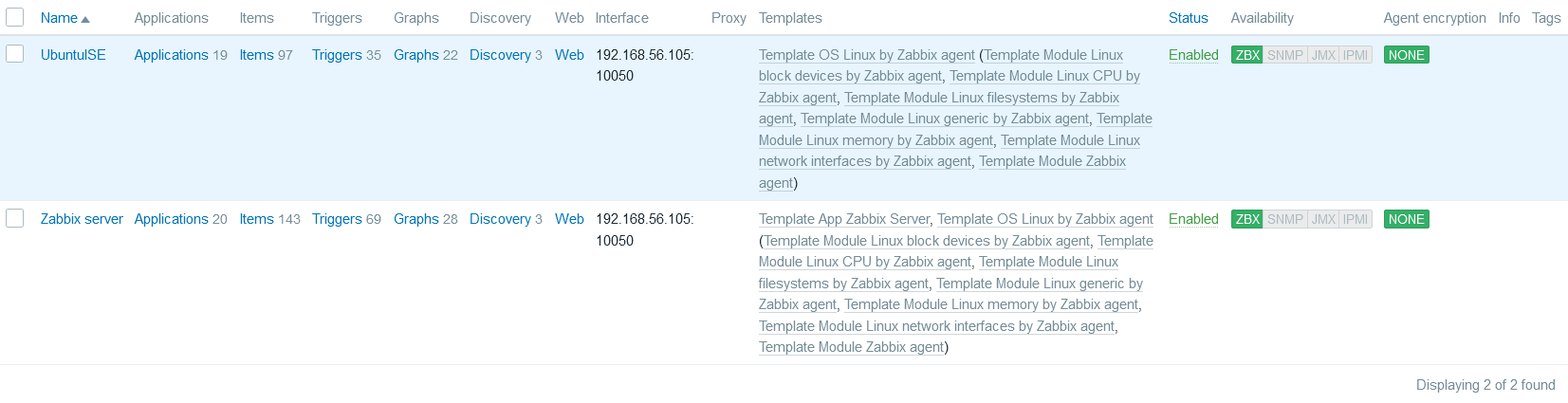
Vamos a añadir un nuevo host, nuestra máquina virtual Ubuntu (agente).

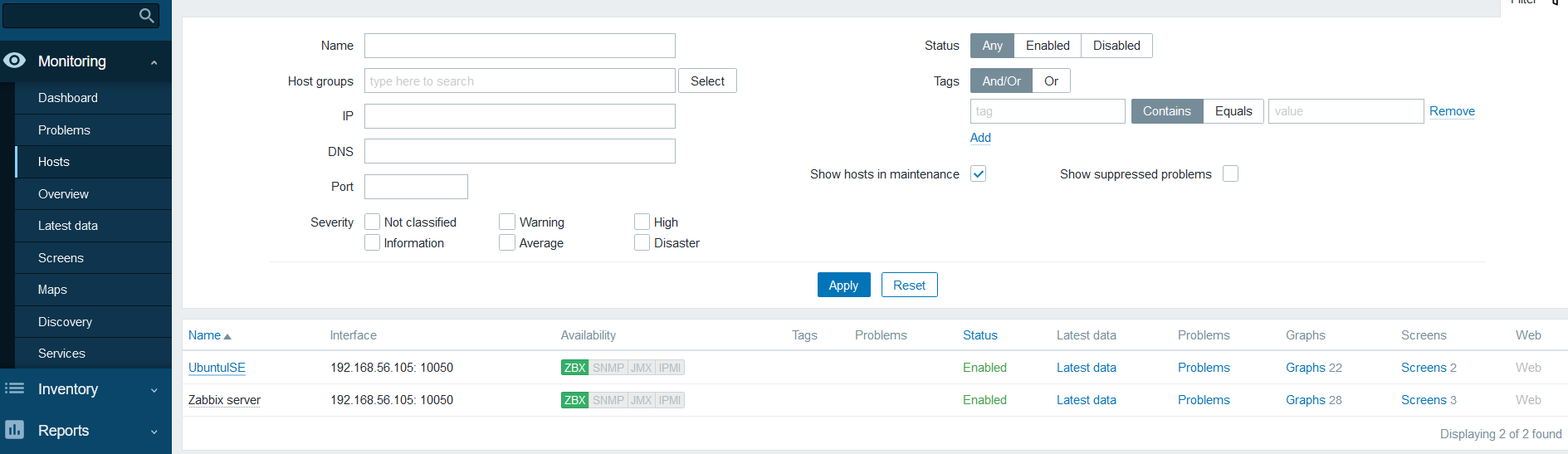
No debemos olvidar incluir la IP en el fichero de configuración del agente de Zabbix en la máquina con Ubuntu Server como se indica en una nota en el manual de instalación, ya que tenemos Zabbix server en la misma máquina que Zabbix agent. Entonces, modificamos con ***vi /etc/zabbix/zabbix\_agentd.conf***

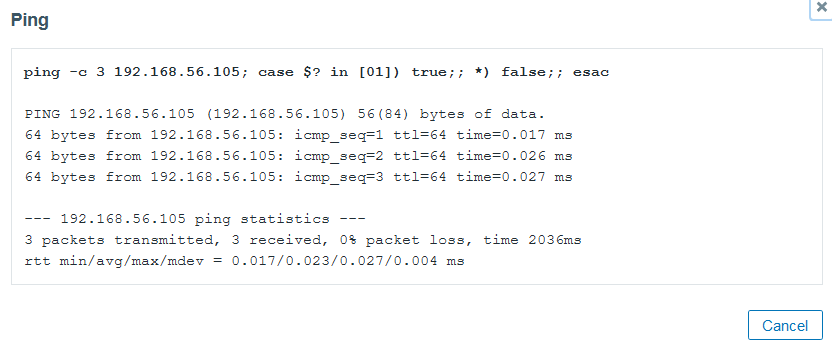
Aplicar con: ***systemctl restart zabbix-server zabbix-agent apache2*** y ***systemctl enable zabbix-server zabbix-agent apache2***

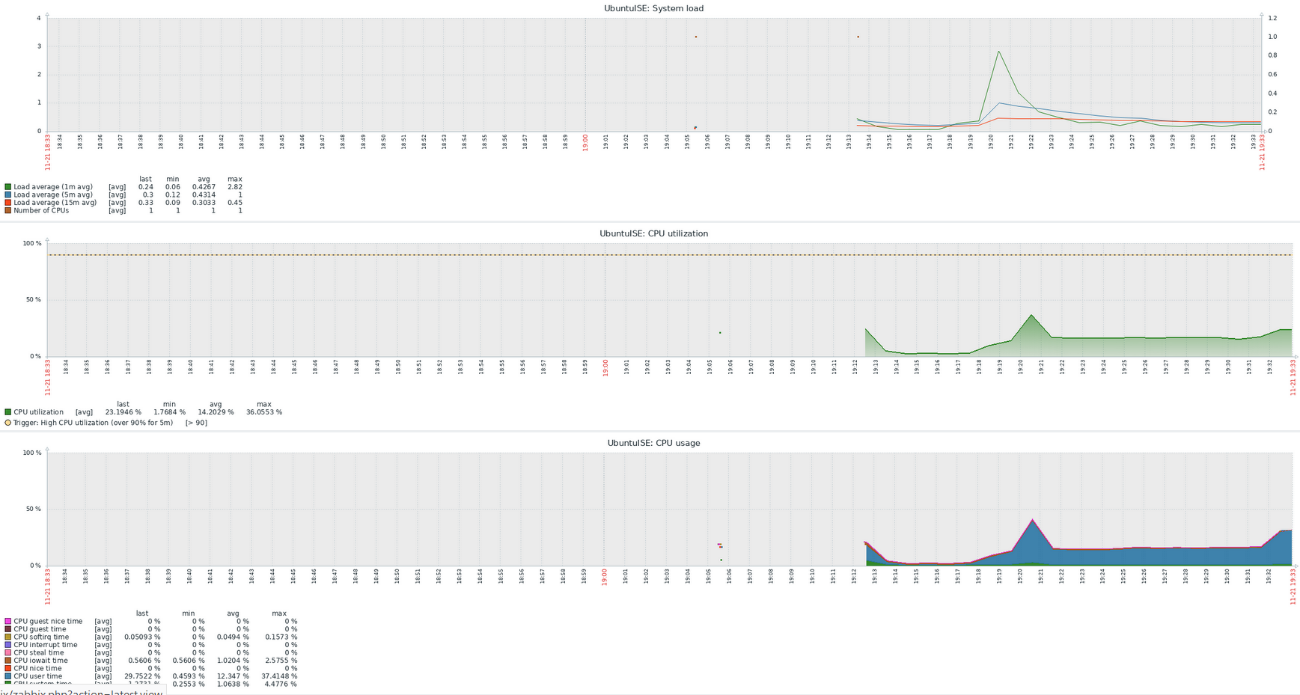
Nota: Asegurarse de que los puertos 10050 y 10051 están abiertos con ***ufw allow***.

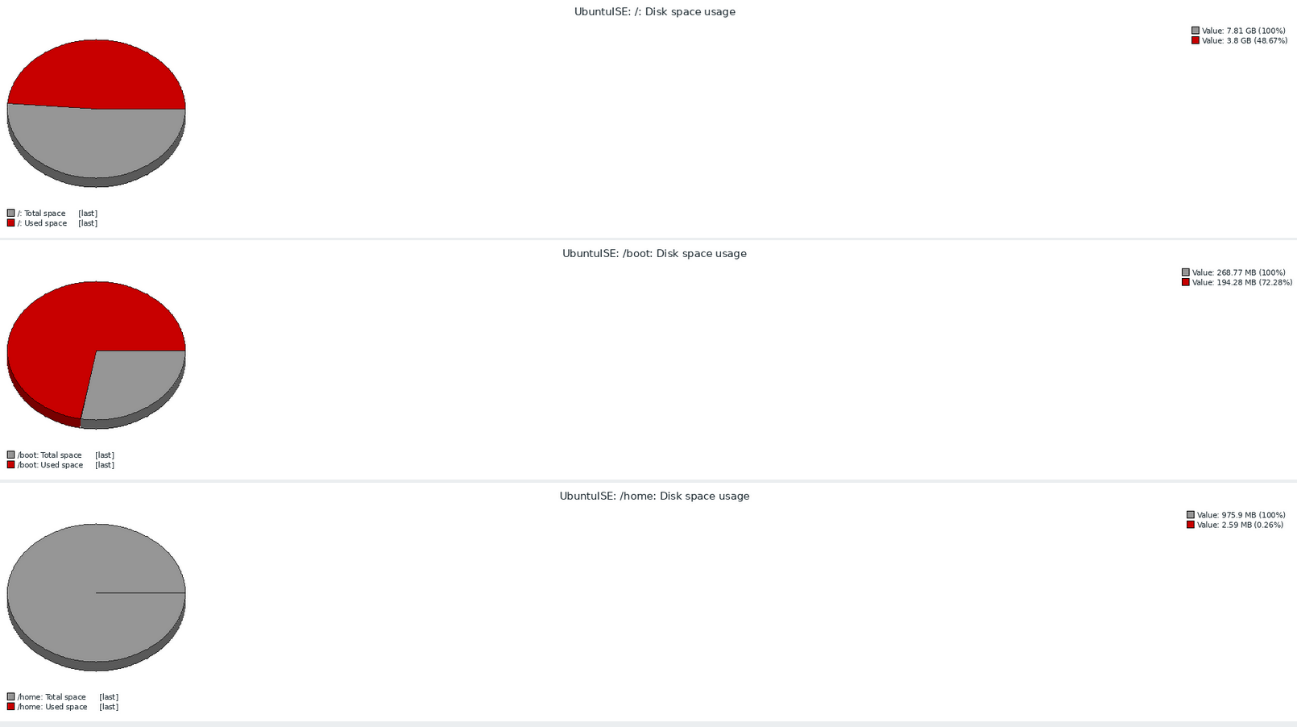
También debemos añadirle la plantilla para los agentes de Zabbix con sistema operativo Linux que incluye la monitorización de CPU, memoria, sistema de archivos, etc lo que nos permitirá poder ver ciertos gráficos e información. Esta plantilla ya está creada.



En esta sección:

Si pulsamos sobre el nombre podemos hacer ping al nuevo host añadido:

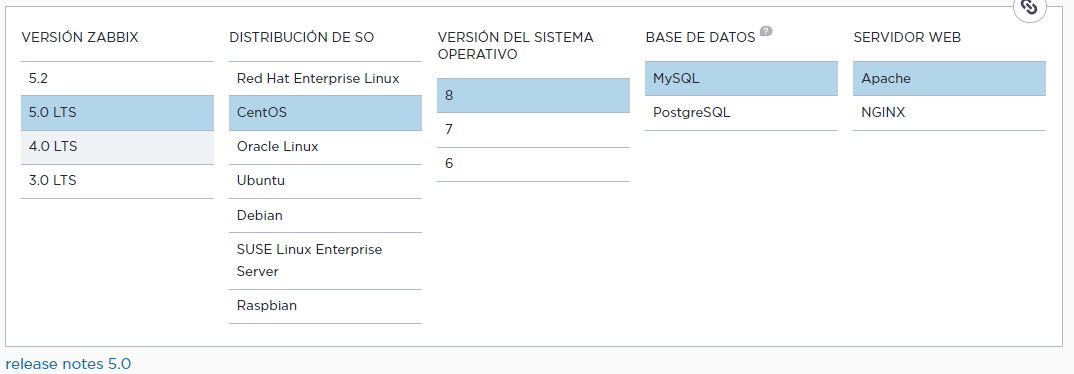
Ya podemos ver ciertas gráficas que se van generando recopilando los datos de nuestra máquina en la sección *Monitoring/Host* y pulsando en la máquina que queramos la opción de *graphs* (estos gráficos también son editables desde la sección de configuración de cada host), por ejemplo, algunas pruebas:

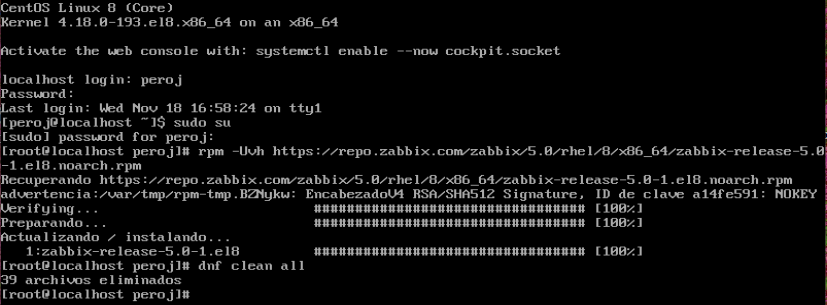
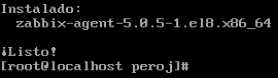
También debemos configurar los *templates* que vienen por defecto con Zabbix para monitorizar los servicios SSH y HTTP. Pero esto lo haremos después de preparar el agente en CentOS.

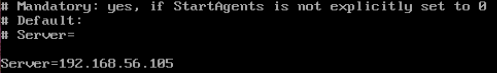
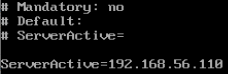
1. ***Instalación y Configuración en CentOS - Zabbix Agent:***

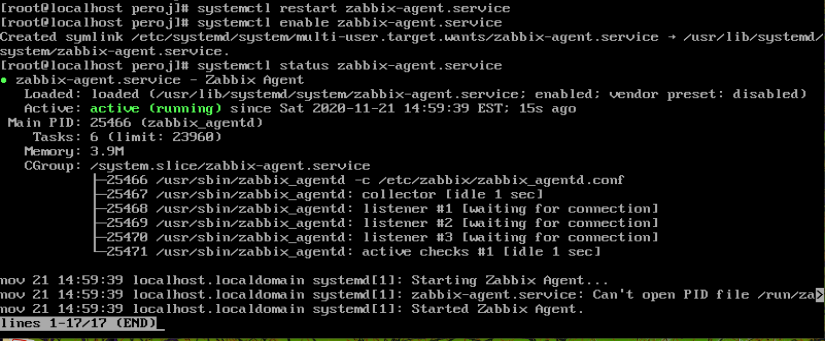
En este caso debemos instalar únicamente el agente de Zabbix en CentOS. Por tanto, marcamos la versión y seguimos los pasos que nos indica ¡a excepción de los relacionados con la base de datos, el servidor apache o zabbix-server!

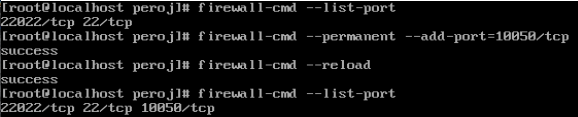
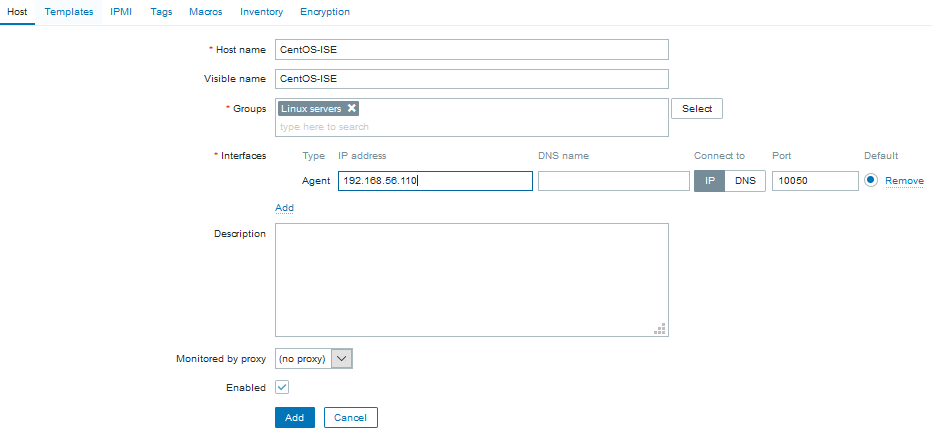
<https://www.zabbix.com/la/download?zabbix=5.0&os_distribution=centos&os_version=8&db=mysql&ws=apache>

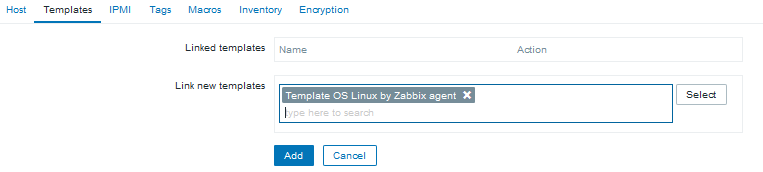


1. Instalar el repositorio de Zabbix con ***rpm -Uvh*** [***https://repo.zabbix.com/zabbix/5.0/rhel/8/x86\_64/zabbix-release-5.0-1.el8.noarch.rpm***](https://repo.zabbix.com/zabbix/5.0/rhel/8/x86_64/zabbix-release-5.0-1.el8.noarch.rpm)y ***dnf clean all***
2. Instalar el agente de Zabbix con ***dnf install zabbix-agent***
3. Configuración del agente de Zabbix en CentOS:

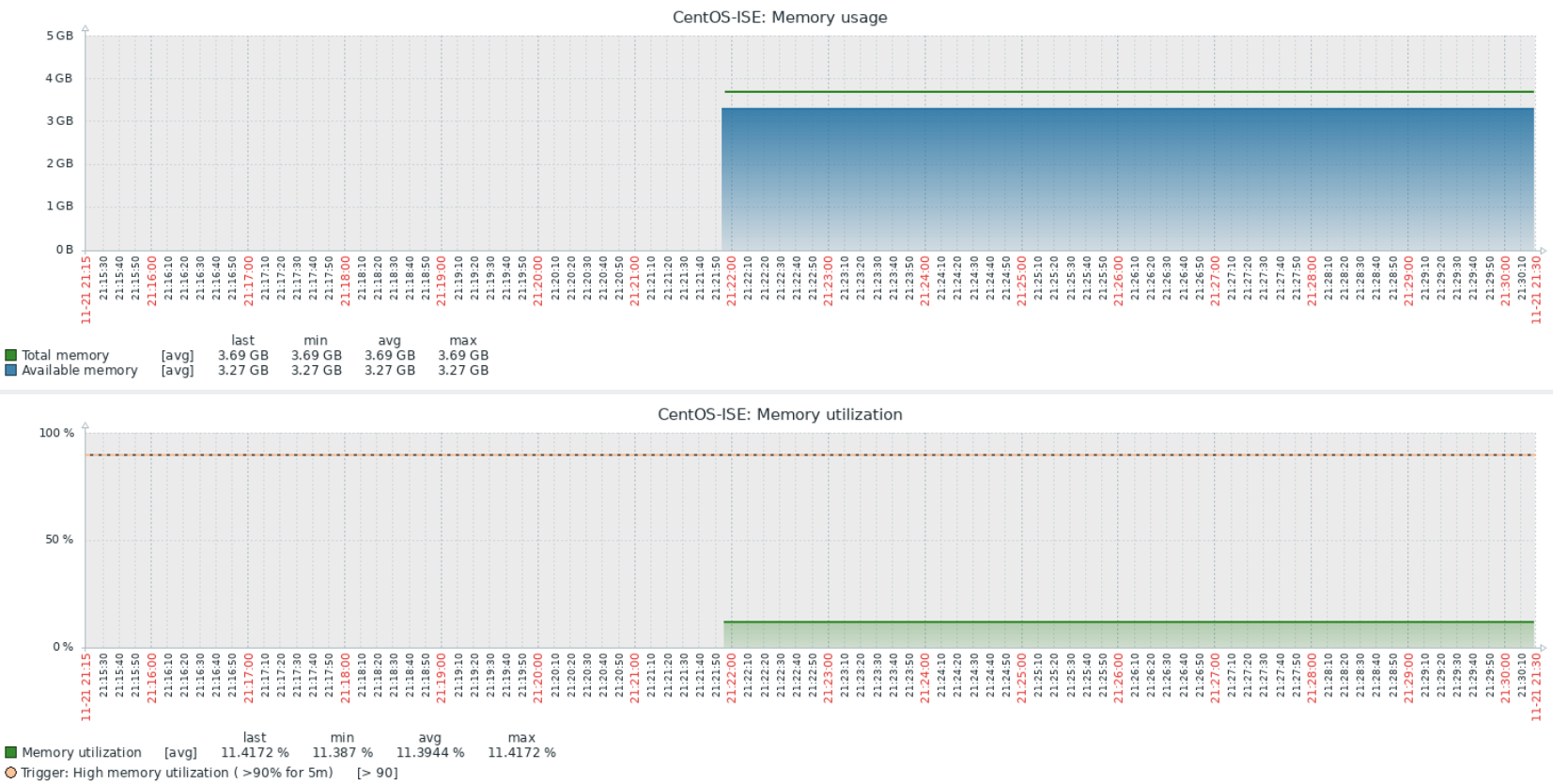
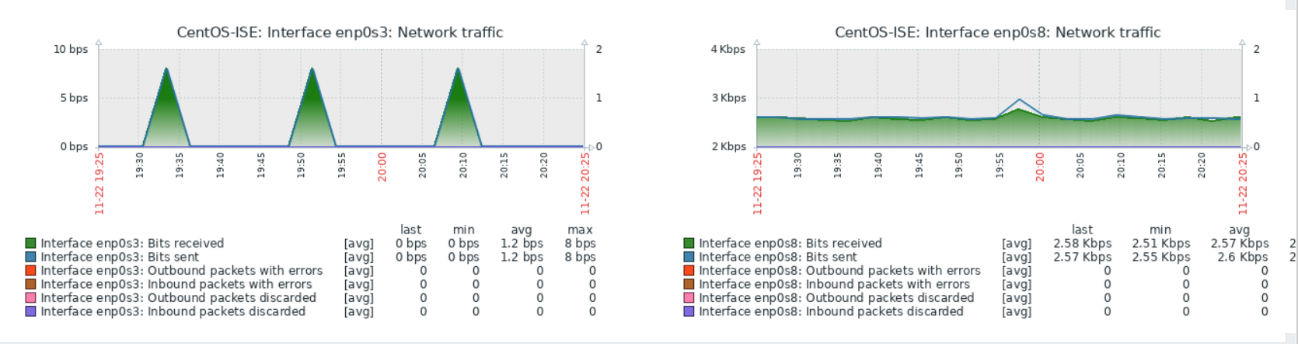
Modificamos el fichero ***/etc/zabbix/zabbix-agentd.conf*** como hicimos para configurarlo en Ubuntu, indicando la dirección IP del servidor de Zabbix en los parámetros “Server” y “ActiveServer”.

Aplicamos los cambios y comprobamos:

1. Configuración de los puertos con ***firewall-cmd*.** Avisamos al cortafuegos de que permita conexiones por ese puerto con ***sudo firewall-cmd --permanent --add-port=10050/tcp*** y recargamos las reglas con ***sudo firewall-cmd --reload***.
2. Creamos un host nuevo desde el *frontend* de Zabbix para CentOS (agente), como hicimos anteriormente con Ubuntu.

Añadimos las plantillas (en un principio solo añadimos la misma que hemos añadido en Ubuntu para comprobar que funciona correctamente):

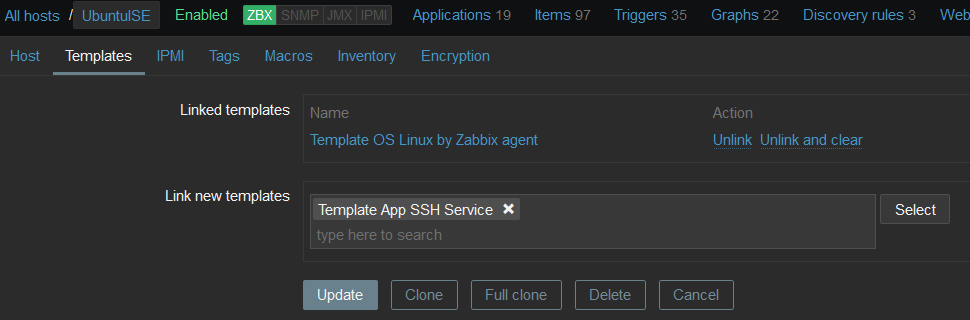
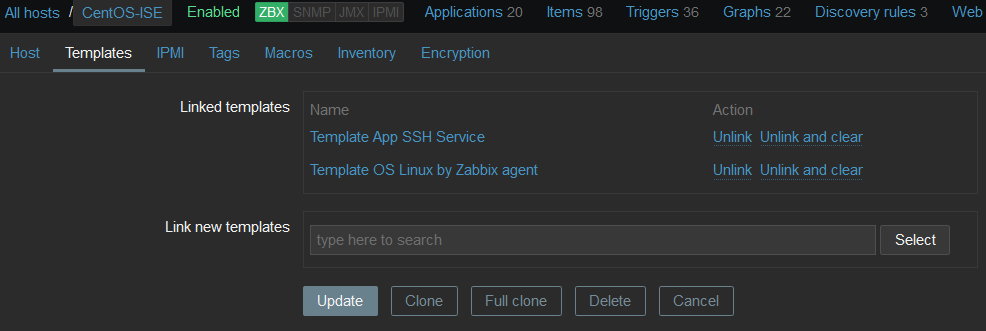
Comprobamos que ya está activo y monitorizando:



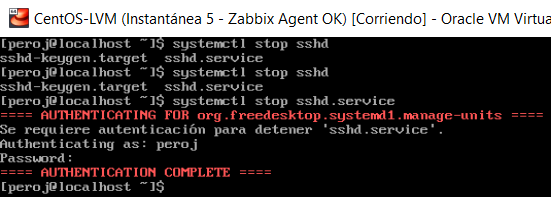
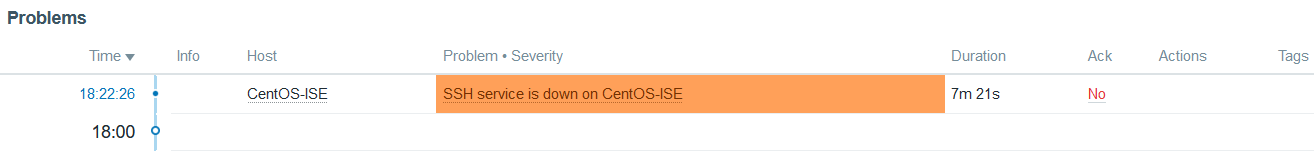
1. ***Monitorización de servicios SSH:***

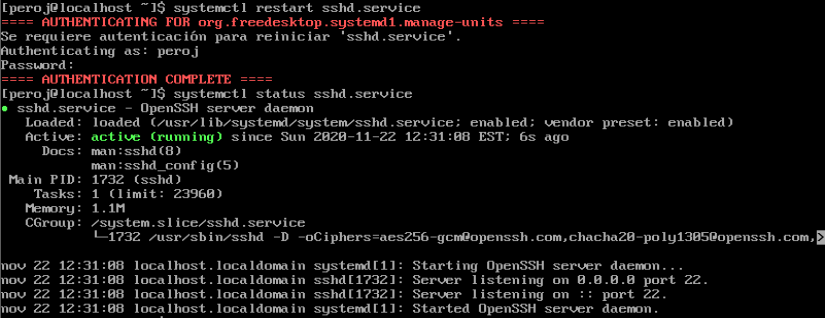
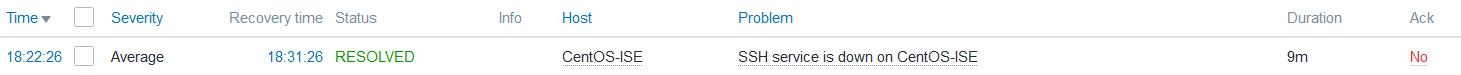
Consultamos el manual: <https://www.zabbix.com/documentation/5.0/manual/config/templates>

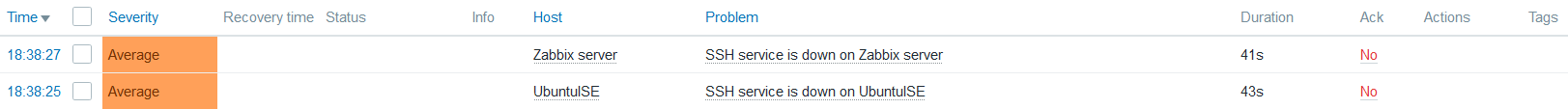
Tenemos la plantilla *Template App SSH Service* que hará un checkeo simple por el puerto por defecto (22, cambiado previamente).

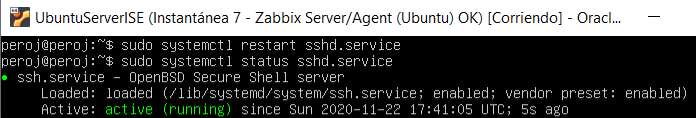
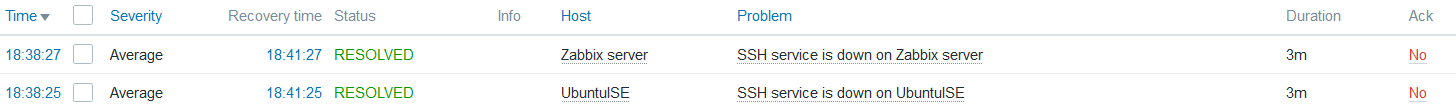
Para monitorizar el servicio SSH de las dos máquinas debemos irnos al *frontend* y añadirle a cada una de ellas el *template* siguiente:

También se lo añadimos al servidor. Esto creará un nuevo *Trigger* que se disparará cada minuto y controlará si el servicio está activo o no:

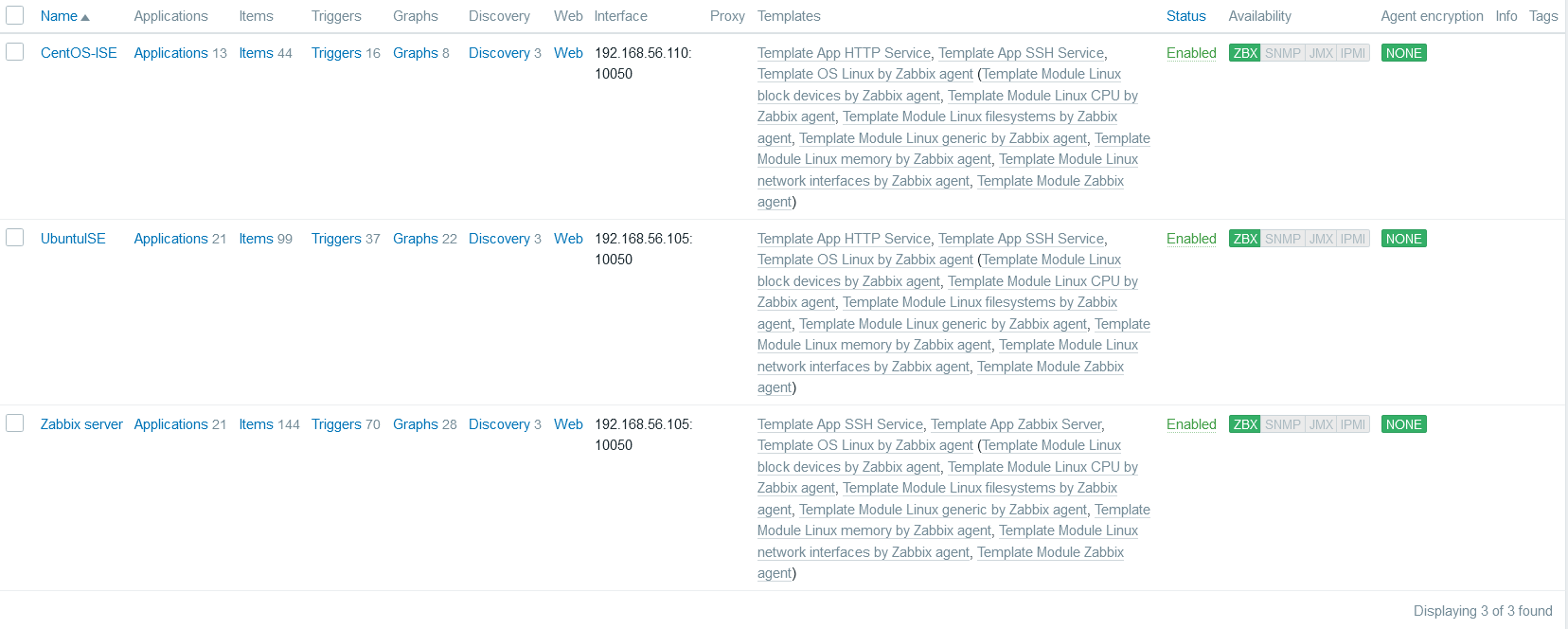
Comprobamos su funcionamiento desactivando el servicio SSH, por ejemplo, en CentOS y vemos como salta el problema:

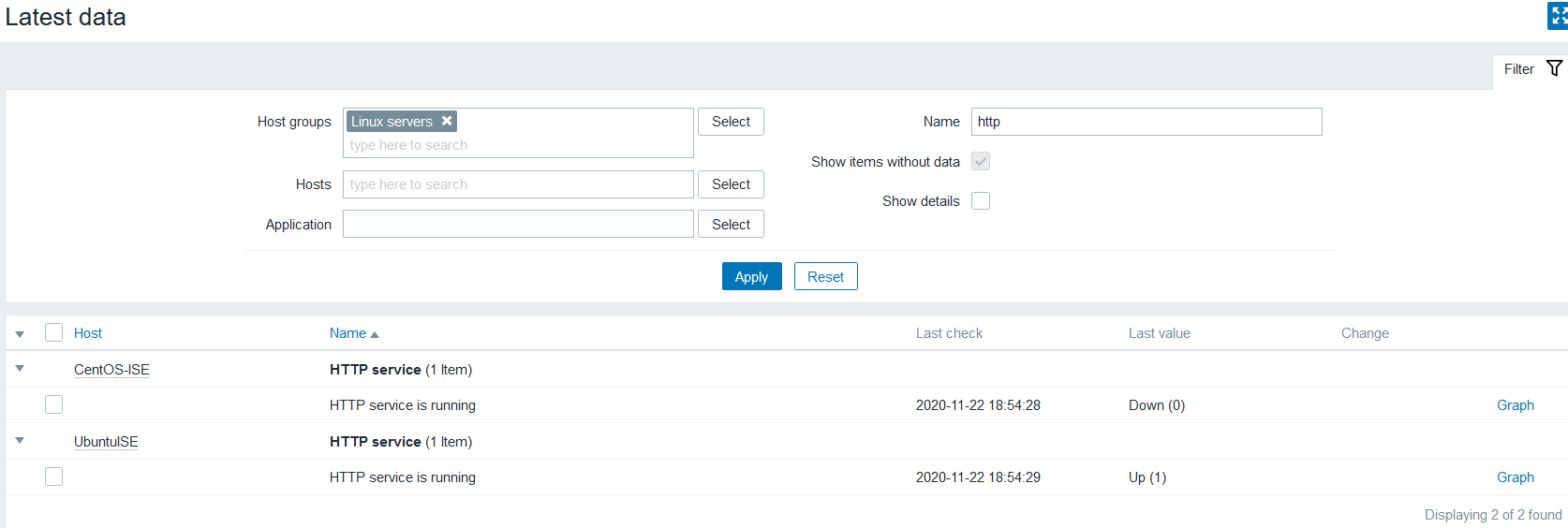
Lo volvemos a activar con ***systemctl restart sshd.service*** y vemos como se corrige el error:

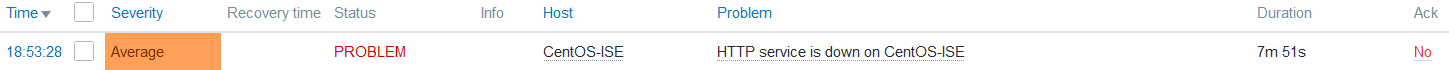
Comprobamos en UbuntuServer:

Reiniciamos el servicio:

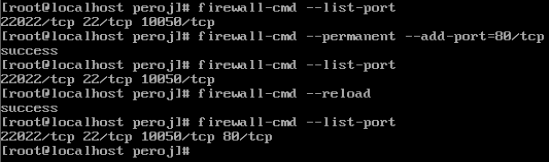
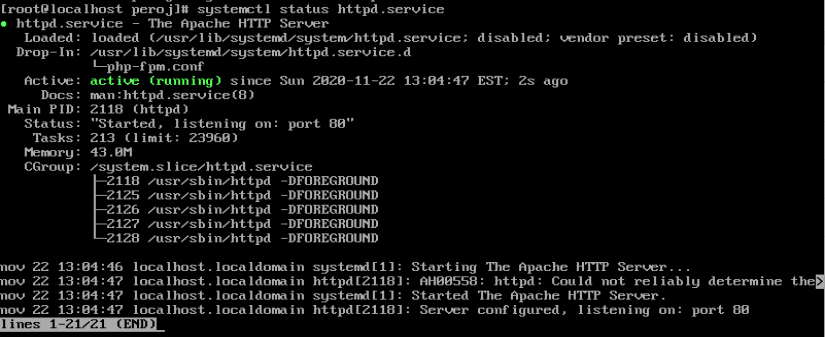
1. ***Monitorización de servicios HTTP:***

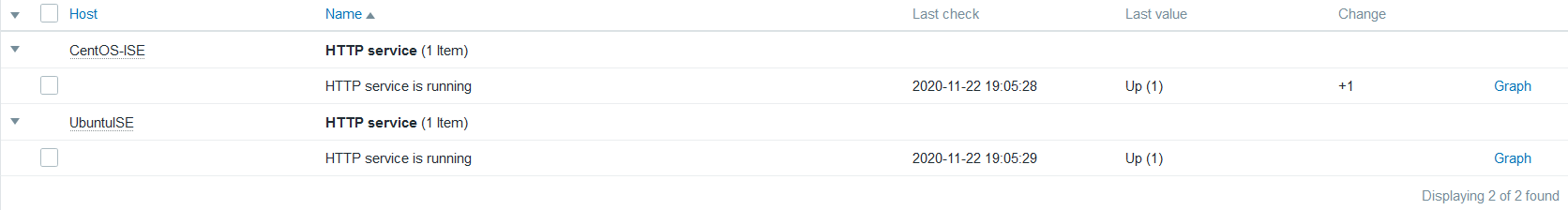
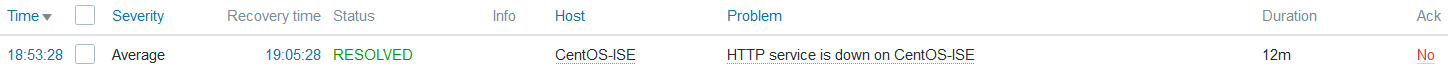
Pues en este caso usaremos la plantilla *Template App HTTP service* y se la añadimos a los host de igual manera.

Comprobamos en los últimos datos recibidos por Zabbix y filtramos por nombre:



Vemos que ya se está monitorizando el servicio en las dos máquinas, pero en CentOS aún nos indica que el ultimo valor fue *Down(0)*. Nos aseguramos que en CentOS esté abierto el puerto 80, utilizado por este servicio por defecto y activado el servicio *httpd*.



Y comprobamos que ahora está en *Up(1)*.

¡Todo correcto!