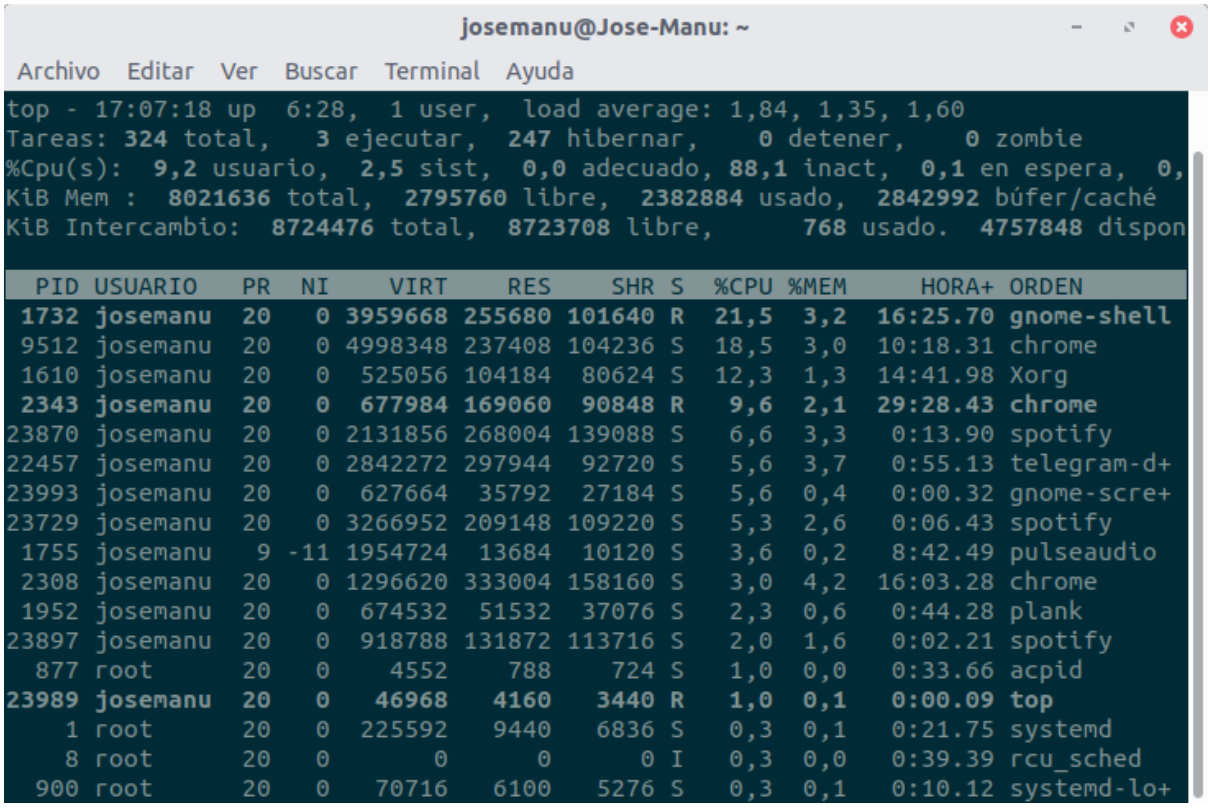
# Lección 2 - P3 ISE

## Comando `top`

Herramienta estandar para monitorizar los recursos del sistema en timepo real. Sus datos vienen dados por el sistema `/proc/` (`/proc/loadvg` para la carga media del sistema por ejemplo)



Instalación → Linux: `apt install top` CentOS: `yum install top`




ejemplo de ejecución

Podemos ver con `top` varias métricas,la primera: la hora de la ejecución de top; la segunda, el tiempo que lleva la máquina encendida, `load average` que es un estándar para mostrar cual es la carga del sistema. Muestra 3 números que por orden son:

- Carga durante el último minuto
- Carga durante los últimos 5 minutos
- Carga durante los últimos 15 minutos

Estos números realmente son una media exponencial del uso de la CPU.

> 💡 La interpretación de estos números sería: el número entero los procesos que se han podido ejecutar, y los decimales la cantidad de procesos que al restar el `load average` a las CPUs que tenemos nos mostrarían los procesos encolados. Veamos un ejemplo: si tenemos un unico procesador, solo podríamos tener un proceso corriendo, por tanto un `load average` de 2 nos indica que 1(2-1) procesos estan encolados de media. Si obtubiermaos un `load average` de 0.40 nos indica que de media, nuestra CPU ha estado ociosa un 60% por ciento del tiempo (1 - 0.40).

Basicamente, a media es la longitud de la cola de procesos que estan esperando para entrar a CPU.

Un sistema cargado más del 75% nos supone un importante riesgo paraa los componentes.

Otros comandos:

`uptime` → da la linea de arriba de top (`load average`)

`vmstat` → estadísticas de memoria virtual (también disponible en `top`)

`free` → estadísticas de memoria principal y swap (también disponible en `top`)

`stress` es un programa que sirve para simular carga en la máquina

**Vamos a analizar ahora lo que top nos muestra:**

`task` o `tareas`: procesos totales y su estado

`sleeping` o `hibernar` : cuando el sistema esta esperando una entrada o salida

`stopped` o `deterner`: procesos parados, orden explícita del usuario

`zombie`: un proceso padre genera procesos hijos y mueren dando un resultado ya sea satisfactorio o erróneo, si no devuelve esto el proceso se queda zombie, consume recursos pero no hace nadda.

Siguiente linea: `% de cpu`

`user` o `usuario`: porcentaje que pasa la CPU en tareas de usuario

`system` o `sistema`: porcentaje de CPU realizando tareas propias del sistema

`nice` o `adecuado`: porcentaje de procesos `nice` que son procesos que nuestro planificador de tareas ha decidido darle un impulso en la cola para que entren a CPU

`iddle` o `inactivo`: procesos que realmente no hae nada pero mantiene la CPU activa, por ejemplo, los procesos de arranque del sistema que son bucles infinitos para que el sistema pueda funcionar

> 💡 Esto consume energía pero en la actualidad un gran grupo de ingenieros estan estudiando como podrían evitar este pequeño consumo

`wait` o `en espera`: el tiempo que pasan los procesos en entrada/salida equivalente a sleeping

`hi`: el tiempo que pasa la CPU procesando interrupciones hardware

`si`: el tiempo que pasa la CPU en tareas `softirq` que son interrupciones software, es decir, cuando el usuario manda una terea al planificador de tareas

`st`: stolen time porcentaje de cpu que un hipervisor le roba a tu CPU, solo disponible en los hipervisores nativos de Linux.

Siguientes y últimas líenas: info de memoria, lo da en k pero con la tecla `e` cambia entre tooooooooooodas las unidades de memoria.

`total`: toda la memoria principal de la que disponemos

`libre`: la memoria que actualmente no estamos usando

`usado`: la porción de memoria que está siendo usada actualmente

`buff/cache`: es un espacio de memoria usado para entrada salida, pero no está reservado de forma infinita, si no que, a veces se "roba" parte de este espacio para procesos (es decir, se puede considerar como libre)

> 💡 si pulsas interrogacion(`?`) sale un ayuda (como el man). Ahí nos dan teclas útiles como por ejemplo `H`, que conmuta entre threads y tasks

La memoria `swap` o `intercambio` tiene los mismos parámetros que la principl, a diferencia de la medida final, la cual no está en todas las distros Linux y es una aproximacion de memoria que un proceso puede demandar ahora mismo y se le puede dar.

Debajo de todo esto estan los procesos que más estan consumiendo, se pueden ordenar como queramos pero por defecto esta por id del proceso.

▼ Algunas teclas útiles:

`P` → ordena por consumo de cpu

`k` → matar procesos

`e` → cambia unidades

▼ Columnas de procesos

`pr`: prioridad

`ni`: nice, explicado anteriormente

`virt`: memoria virtual total

`res`: memoria real

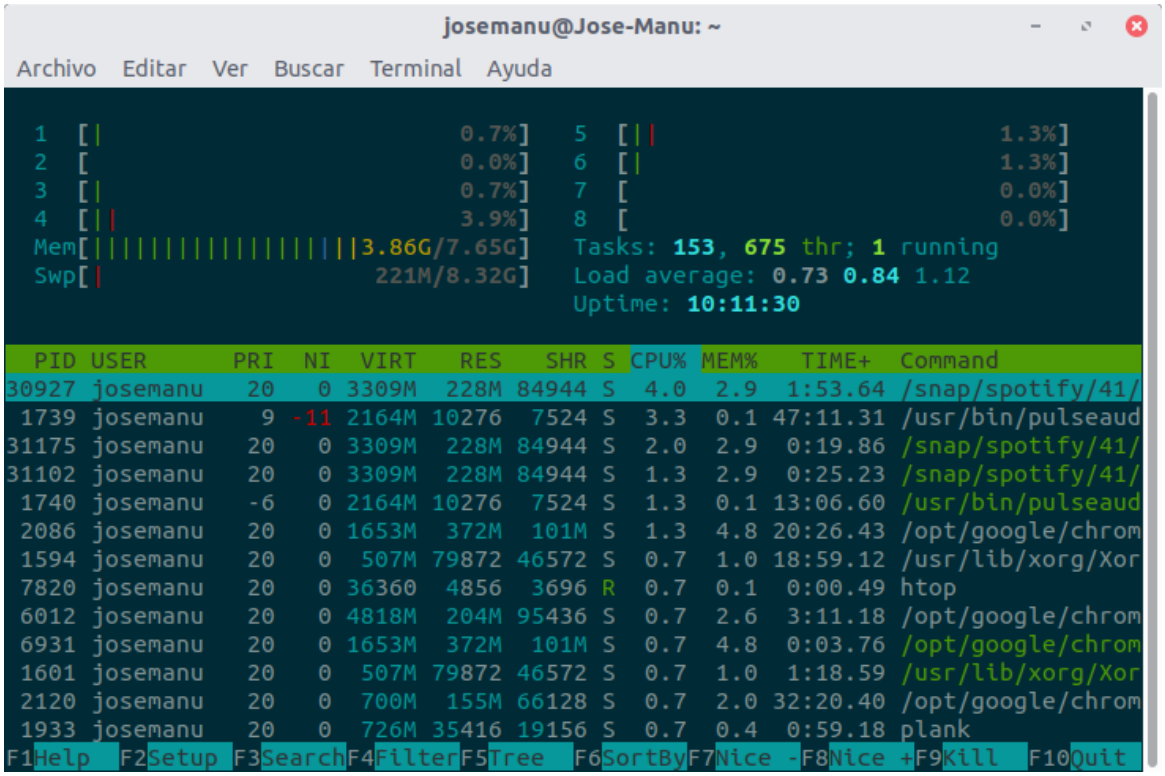
`shr`: espacio de memoria compartio

`s`: estado

**Otras versiones de top, con características añadidas:**

`htop` (t`op` pero bonito) → te pinta las cosas graficas y bonitas

para instalar → `apt install htop`

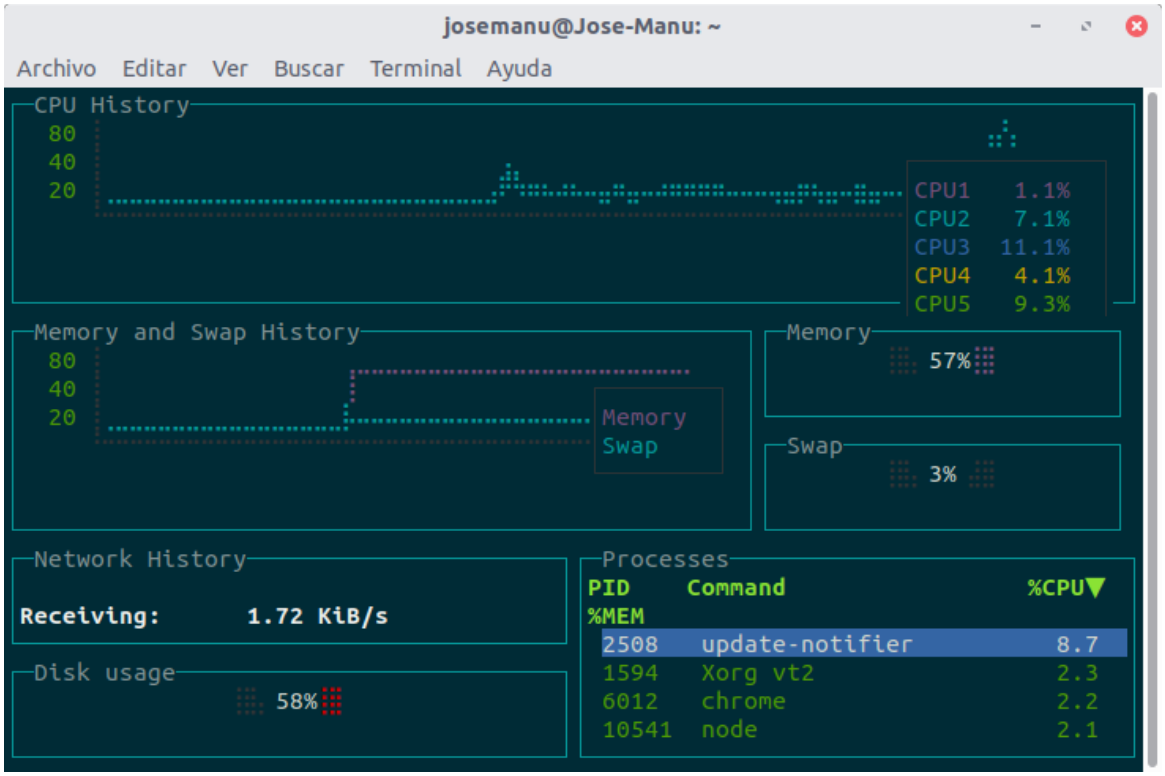



`gtop` (como `top` pero todavia mejor)

depende de node, el cual es un entorno en tiiempo de ejecución basado en java.



Como depende de node, instalaremos node en primer lugar → `apt install nodejs`, posteriormente `npm`, el cual es un gestor de paquetes para obtener proyectos open-source de Node.js. Una vez hecho esto lanzamos `npm install gtop` y listo




una preciosidad, vaya



Referencia chula del profe, nos muestra una web en la que hace una comparativa de recursos que usan diferentes lenguajes de programación

Hemos visto que `top` es muy útil y tiene un montón de hermanitos geniales y muy visuales. Peeeeeeeeeeeeeeeeero `top` no sirve para una red de server y menos para hacer un histórico :(

Tener un histórico nos sirve para ver lo que podemos mejorar dependiendo del tráfico de la web, tendecia del uso de los recursos. Vamos sirve pa que no nos echen del trabajo si somos administradores de un servidor: tenemos que anticiparnos

ejemplos de monitores de servidores: munin, prometheus, este ultimo + grafana que es un paquete que mejora el análisis de los graficos, + influxdb (PIG, sería su acrónimo) que es una base datos to pollua para guardar info. Pero nosotros vamos a usar `zabbix`.

Nos encontramos con dos estrategis en esto de monitorizar:

- pull: analizamos 5 equipos por ejemplo, va 1 por 1 preguntándole sus variables (más antiguo este método) aquí es el server el que toma iniciativa y pide info
- push: lo iniciativa ahora la toman los equipos, que quieren comunicarse con el servidor central (zabbix por ej) cuando estos quieren ser monitoreados.

¿Qué es mejor? antes tol mundo usaba pull que era lo único que habia, despues se hizo famoso push porque dependia del cliente ya que comunica cuando tiene algo reseñable, esto carga menos el servidor. Push tiene un incoveniente y es que pierdes el control de cuando te llega la informacion (servidor central puede encontrarse saturado)

## Zabbix

Es un sofware de código abierto, gratuito y dispone de una  arquitectura para montar un monitoreo de una red de servidores de modo muuuuuuuuy sencillo. (~~realmente no es tan sencillo xd~~)

Admite pull y push. Por defecto pull, que es lo que vamos a usar.

Es un software que permite monitorizar distintos elementos, ya sea bd, routers, servidores, etc

Zabbix es capaz de llamar los elementos conectados y que les de info de como estan, entradas salidas, usuarios dentro, instancias de apache lanzadas, porcentaje de cpu, temperatura, vamos, un chingo de cosas



Version que vamos a usar????? 3.4

Requiere de instalacion LAMP completa

▼ **¿Qué interviene en zabbix?:**

- `server`: máquina que recibe los datos y los guarda en una base de datos (en nuestro ejemplo en mysql) usa base de datos relacional
- `agent`: es lo que va a tener en segundo plano la máqiuna que será monitorizada y es el encargado de tomar las medidas y enviarlas al server. El agente puede almacenar las métricas durante un tiempo y así cuando el server se lo pide y puede enviar to lo que tenga
- `proxy`: no muy importante, si hay muchos equipos un server no puede acceder a todos, esto es una especie de server pequeño que muestrea redes pequeñas locales, entonces el server se comunica con el proxy. Como el tiempo de abrir una conexion tcp es muy grande interesa hacer esto si estamos monitorizando una gran red

## Vamos a hacer nuestro propio servidor Zabbix y a ponerlo a monitorizar. Oleeeeeeeeee

▼ Requisitos y recomendaciones previas para esta manualidad Art Atack:

Tener una instalación completa de LAMP (Linux, Apache, MySQL, php)

Instalación de Zabbix en su versión 3.4 para CentOS y Ubunto (si se quiere otra versión u otro sistema operativo seguir los pasos indicados en su web

En Ubuntu instalaremos el servidor, el frontend para la gestión de la interfaz web, el agente y la herramienta `get` para obtener resultados en el momento que los pidamos.

En CentOS, instalaremos el agente

Se recomienda hacer `ssh` desde nuestra máquina tanto a Ubuntu como a CentOS en dos terminales independientes.

**Comensemos**

1. Nos traemos el paquete de Zabbix 3.4 en Ubuntu con el siguiente comando `wget https://repo.zabbix.com/zabbix/3.4/ubuntu/pool/main/z/zabbix-release/zabbix-release_3.4-1+xenial_all.deb`
2. Instalamos el paquete con la siguiente línea `dpkg -i zabbix-release_3.4-1+xenial_all.deb`
3. Y actualizamos nuestros paquetes con `apt update`

4. Instalamos el server con soporte MySQL que el tipo de base de datos que vamos a usar. (para otras bases de datos mirar su web) `apt install zabbix-server-mysql`
5. Instalamos el frontend `apt install zabbix-frontend-php`
6. Instalamos la opción `get` con la siguiente línea: `apt install zabbix-get`
7. El proceso de instalación ya habría terminado, ahora vamos a crear la base de datos, para ello seguir las siguientes líneas `mysql -uroot -pcontraseña` (en contraseña le asignamos la que gustemos)

```
mysql> create database zabbix character set utf8 collate utf8_bin;
mysql> grant all privileges on zabbix.* to zabbix@localhost identified by 'contraseña';
mysql> quit;
```

8. Vamos a importar el esquema y los datos principales de Zabbix, para ello la compañía ya nos ha preparado un archivo. Simplemente ejecutamos la siguiente línea y actúa la magia `zcat /usr/share/doc/zabbix-server-mysql/create.sql.gz | mysql -uzabbix -p zabbix` nos pedirá la contraseña que introdujimos arriba, se la damos y listo.

9. Vamos a editar el archivo de configuración del servidor, que se localiza en `/etc/zabbix/zabbix_server.conf` con nuestro editor de texto favorito. Basicamente tenemos que descomentar las siguientes opciones y darle los siguientes valores, siempre que hayamos dejado por defecto las parámetros que le pasamos a MySQL, en caso de cambiar alguno aquí también deberiamos de cambiarlo

```
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=nuestracontraseña
```

10. Ha llegado el momento de activar nuestro servidor, para ello ejecutamos los siguientes comandos que se encargaran de este propósito `service zabbix-server start` y `update-rc.d zabbix-server enable`.

11. Y ahora, para activar nuestro frontend y su maravillosa interfaz web reiniciaremos nuestro servicio `http` con el siguiente comando `service apache2 restart`

12. Todo activado, tenemos que configurar nuestro frontend ahora. Vamos a hacerlo. Accedemos su archivo de configuración localizado en `/etc/apache2/conf-enabled/zabbix.conf` con nuestro editor de textos favorito. Aquí vamos a tocar los

parámetros dependiendo de nuestra versión de `php`, en mi caso la 7. Dejaré los parámetros tal que así:

```
<IfModule mod_php7.c>
  php_value max_execution_time 300
  php_value memory_limit 128M
  php_value post_max_size 16M
  php_value upload_max_filesize 2M
  php_value max_input_time 300
  php_value max_input_vars 10000
  php_value always_populate_raw_post_data -1
  php_value date.timezone Europe/Madrid
</IfModule>
```

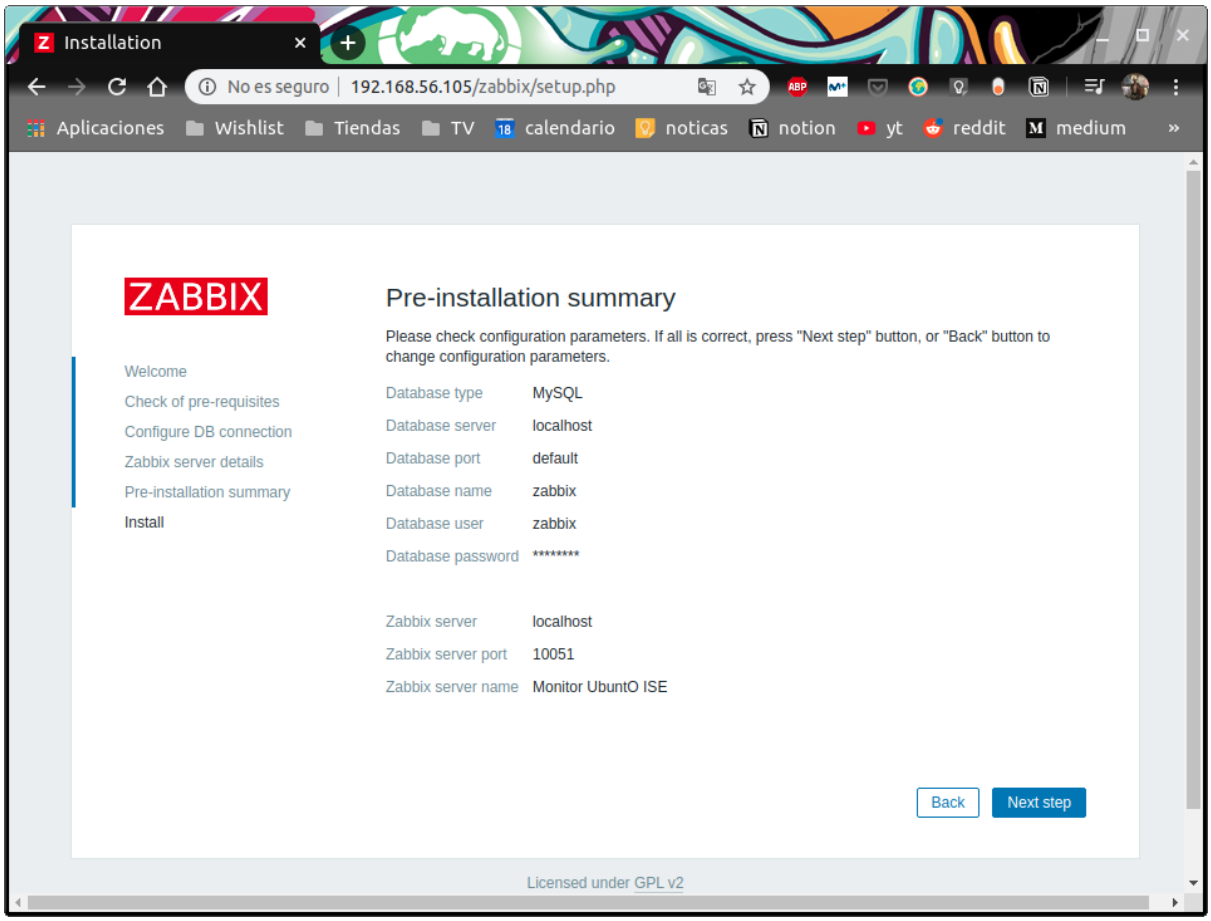
Y listo, tendríamos nuestro frontend configurado, listo para mostrar su interfaz gráfica en nuestro navegador de internet favorito. Vamos a configurarlo accediendo desde nuestro navegador a la dirección `http://ipserver/zabbix`

13. Nos encontramos con la siguiente pantalla

> ! Si no nos muestra esta interfaz puedes ser problema de no tener la instalación LAMP completa u algún error en la creación de la base de datos o configuración del server/frontend. Para asegurarnos podemos echarle un vistazo a los archivos `log` situados dentro de la carpeta `/var/log`

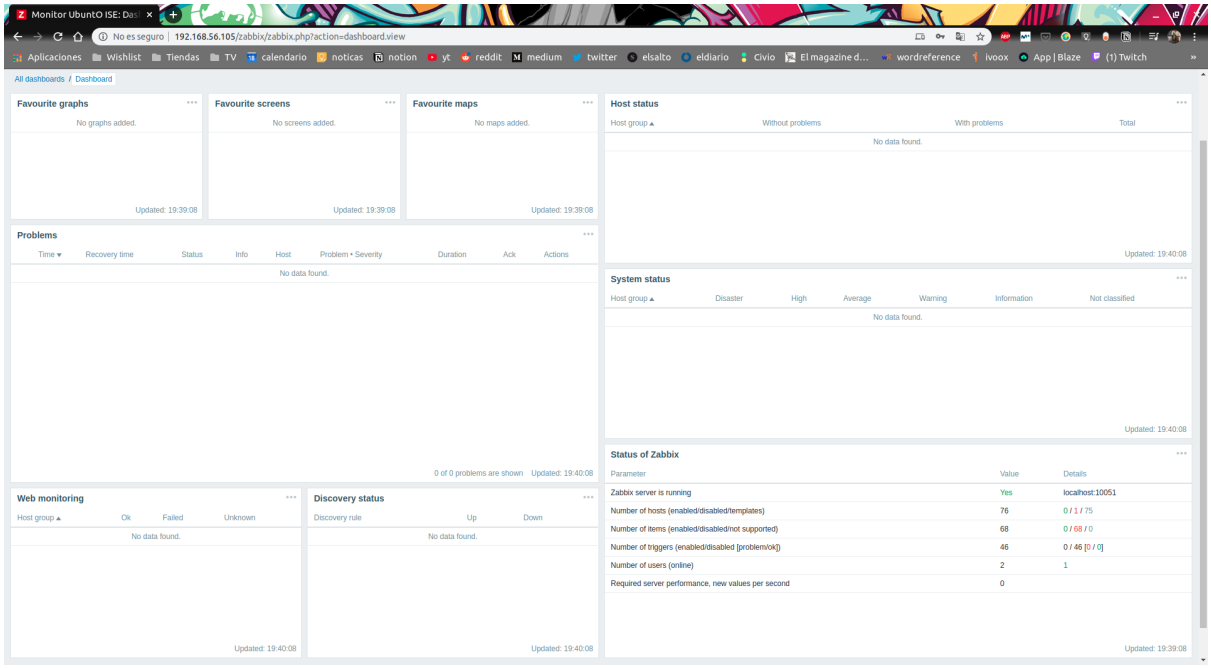
14. Clickamos en `Next step` y vamos rellenando los datos que se nos pide, si no detecta algo en la primera pantalla, referente a `php` accederemos al archivo de configuración del frontend y nos aseguramos de que todo esté correctamente escrito.

15. En el último paso nos quedará algo tal que así




16. Para acceder nos pide un usuario y contraseña, por defecto se trata de `User: Admin Password: zabbix`

17. Si todo ha ido bien, tendremos esta interfaz, si no, repasar los pasos anteriores




18. Vamos a instalar el agente en CentOS, nos cambiamos de SO.

19. Al igual que hicimos con Ubuntu, vamos a traer e intalar en nuestra máquina el paquete de Zabbix 3.4 con la siguiente línea `rpm -ivh https://repo.zabbix.com/zabbix/3.4/rhel/7/x86_64/zabbix-release-3.4-2.el7.noarch.rpm` (`rpm` trae el paquete, `-i` instala)

20. Finalmente instalamos el agente proveniente de nuestro paquete `yum install zabbix-agent`

> ! Podría ser que la instalación nos dé error por el SELinux, el cual es un servicio destinado a la seguridad de nuestro SO, para instalarlo pondremos el modo en permisivo con el comando siguiente: `setenforce 0` esto realmente no soluciona del todo nuestro problema, pero eso lo veremos mas adelante.

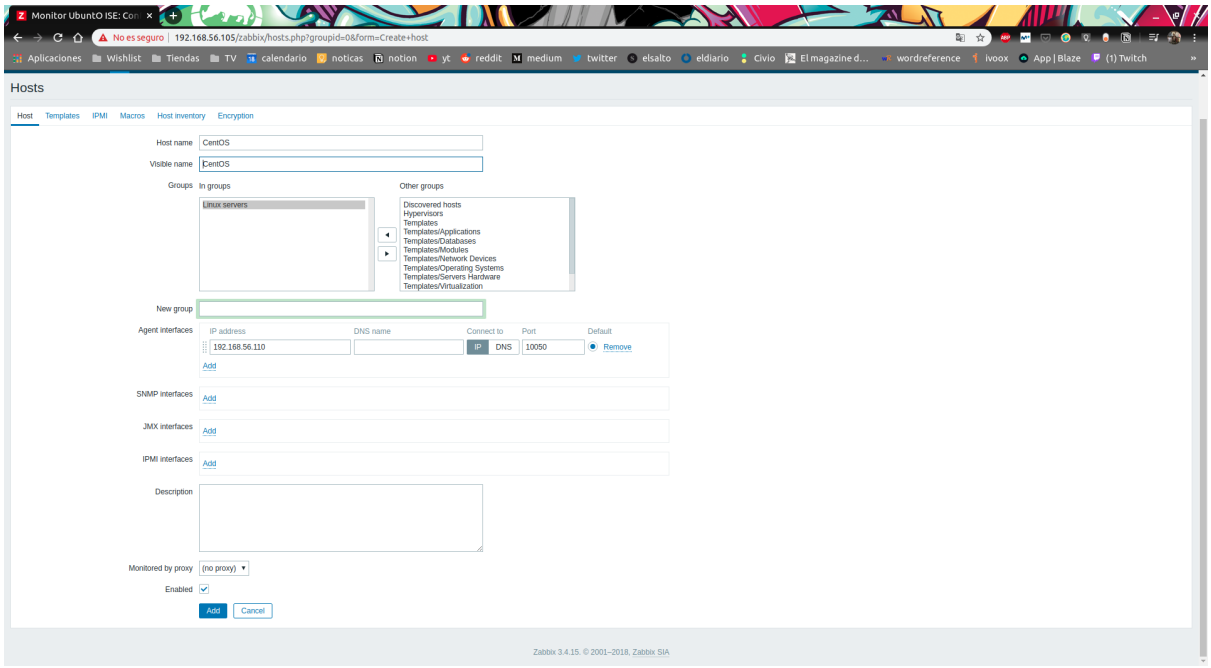
21. Vamos a activar el agente una vez lo tenemos instalado con el comando `service zabbix-agent start` Podría ser que ahora nos de error si no pasó en el paso 20, si es así ejecutamos el comando que expliqué antes.

22. Activamos el puerto de escucha en CentOS (por defecto 10050) con el comando `firewall-cmd ..zone=public --add-por=10050/tcp --permanent`

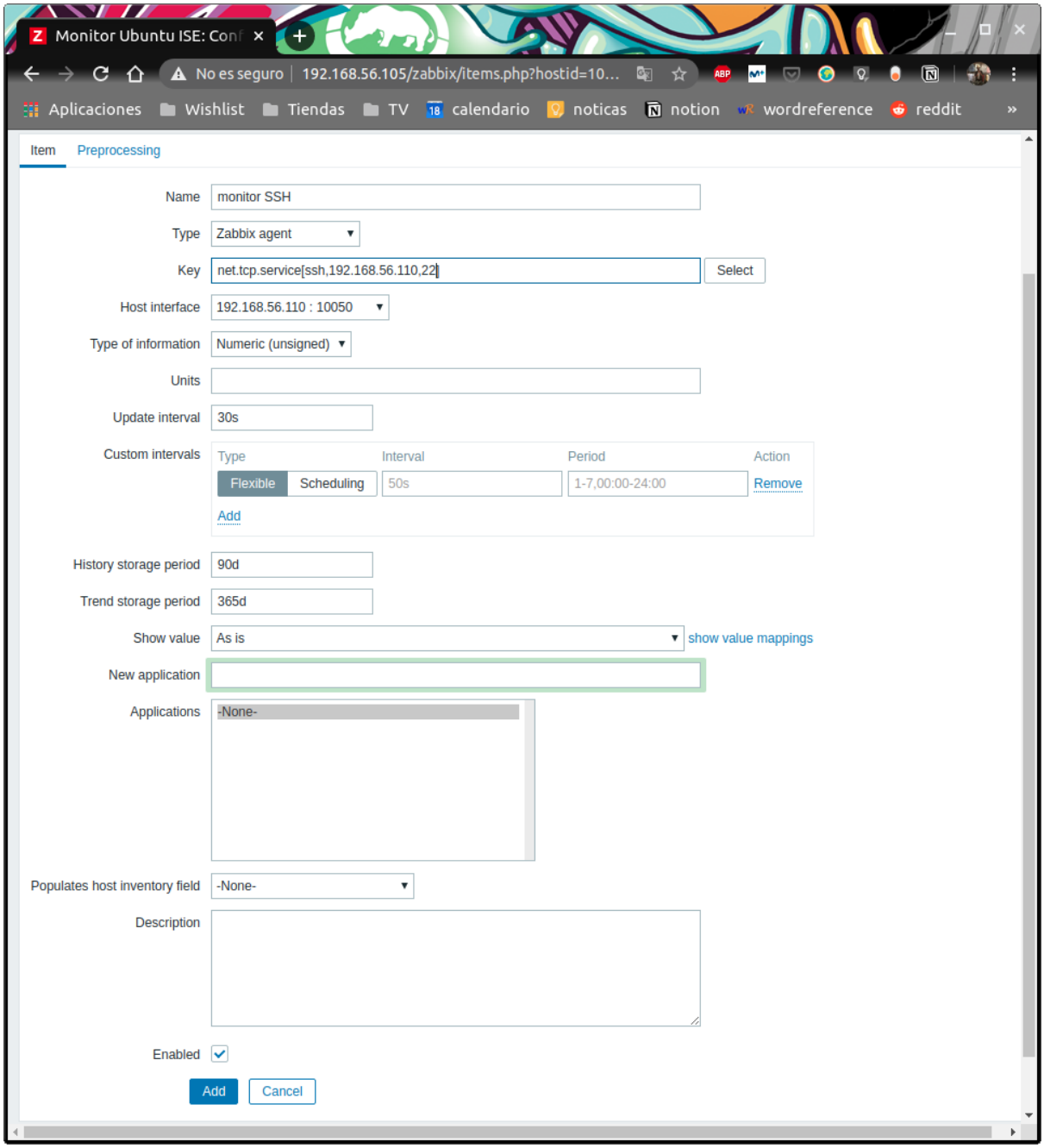
> ! Si nos encontramos con algún problema activamos los puertos `10051` en Ubuntu y CentOS y el `10050` en Ubuntu también

23. Todo listo, vamos para nuestra interfaz web para configurar nuestros monitores de SSH y HTTP en CentOS.

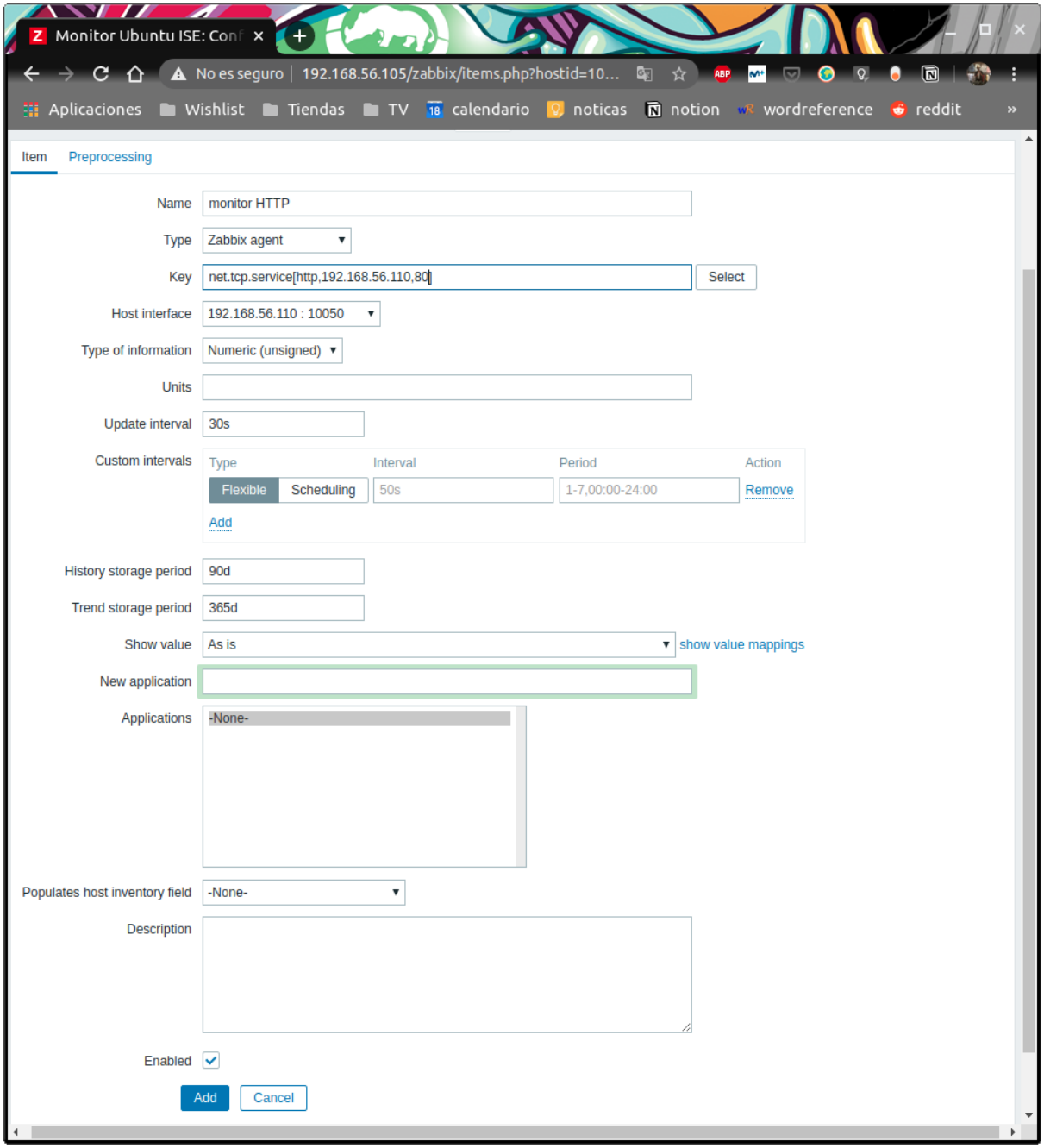
24. En la barra superior, buscamos la opción **configuration**, clickamos, posteriormente se abre un desplegable, aquí click a **hosts**, se abre un nuevo menú, click en **Create new host**, y modificamos los parámetros de este modo, donde ip, será la ip de nuestra máquina a monitorizar:



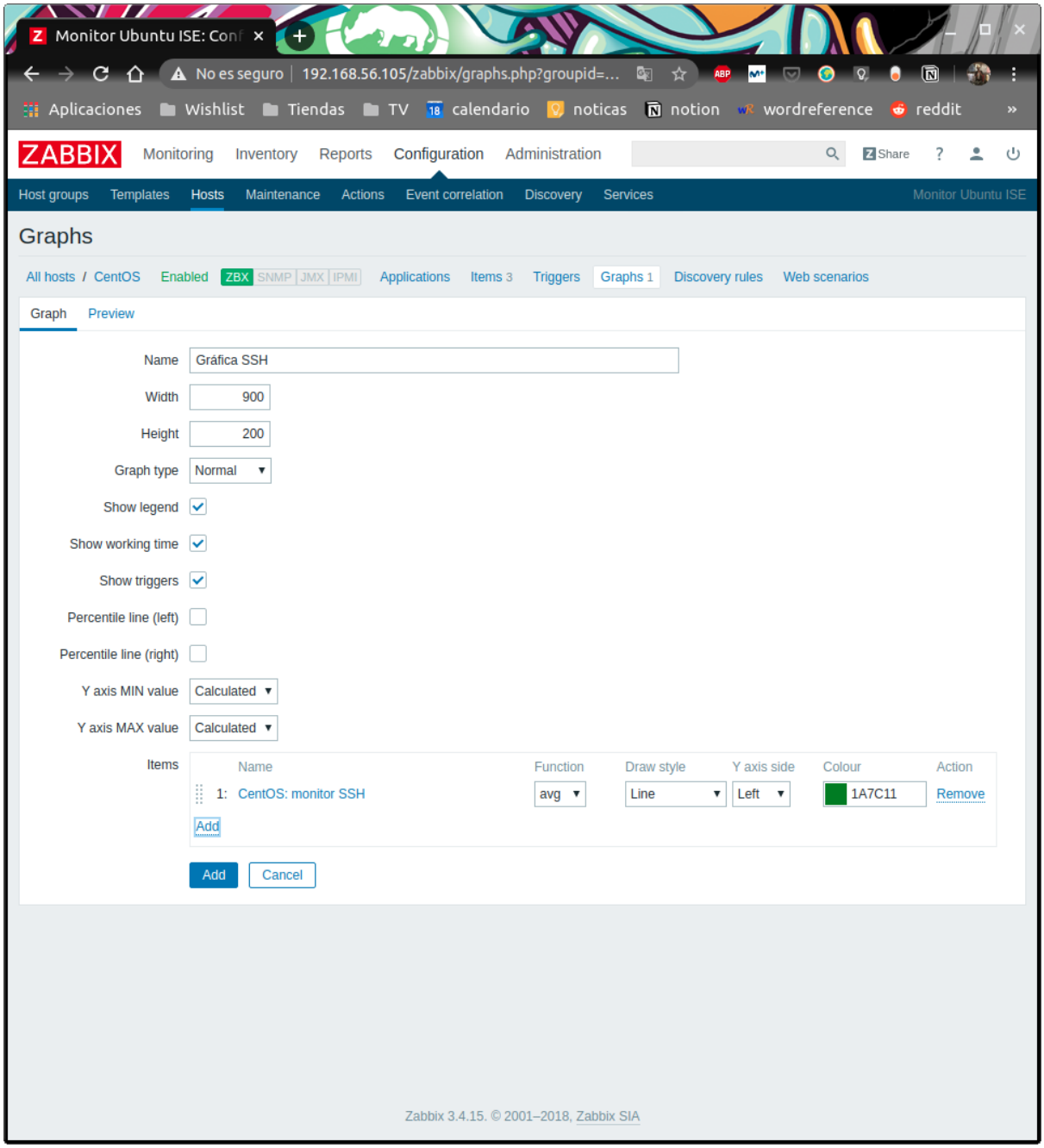

25. Una vez creado el host, vamos a por los servicios, para ello en la línea de **CentOS** damos click en **Items**, posteriormente en **Create item** y elegimos estos parámetros para `ssh`**:**



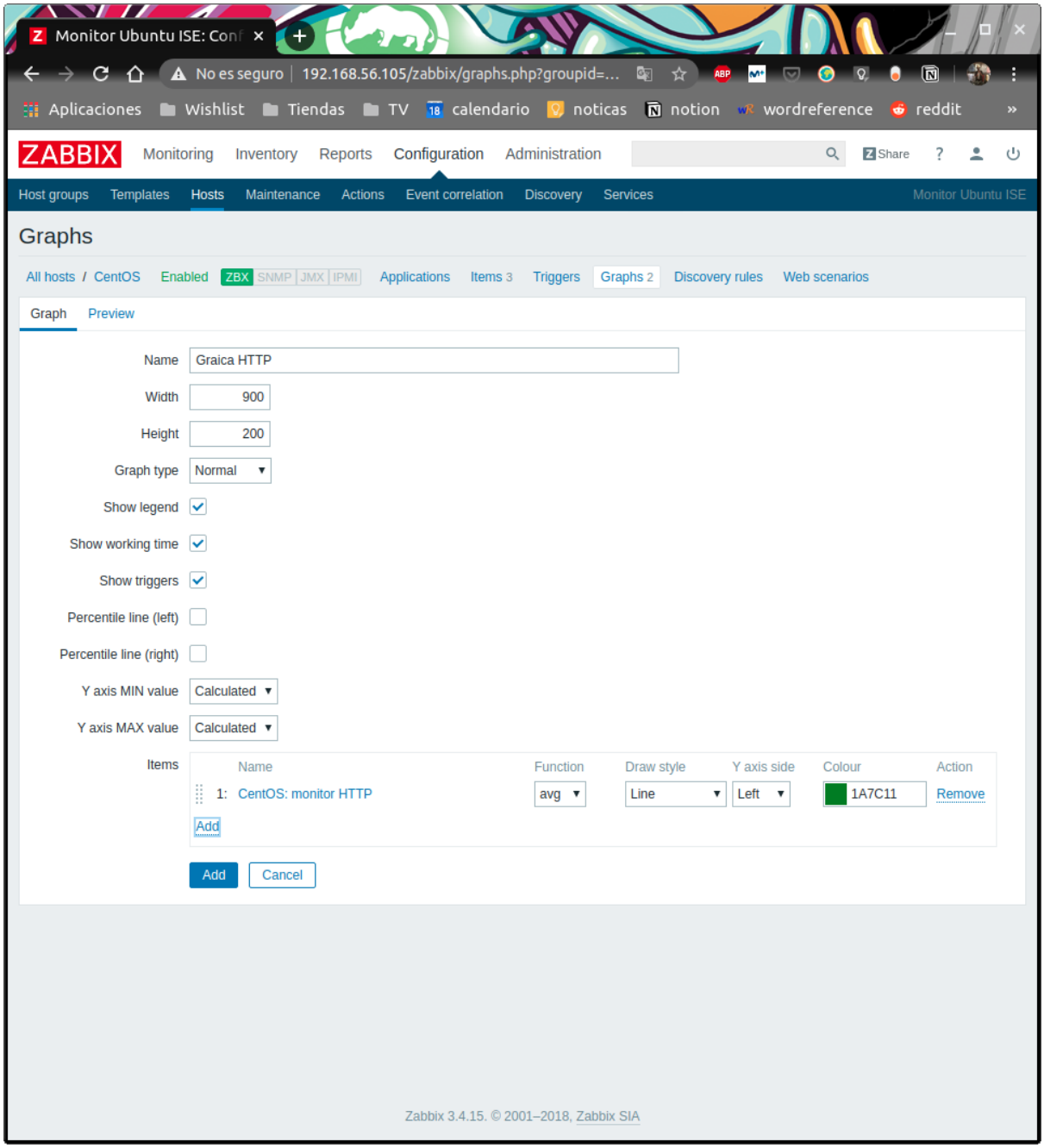

26. Una vez clickada la opción **add**, vamos a crear otro para ver nuestro `http`, para ello repetimos el paso 25, pero con estos nuevos parámetros:




27. Vamos a ver si esto está activo o desactivo mediante una gráfica, ya que será muy visual ver una caída a 0 si deja de funcionar y un 1 si tenemos estos servicios funcionando, para ello, igual que en los pasos anteriores clickamos en **item** ahora vamos a hacerlo en **graphs**. Aquí se abre un nuevo menú, click en **Create graph** y dejamos los parámetros como vemos aquí:




28. Lo mismo con `http`:




29. Para ver las gráficas click en **Monitoring** y posteriormente en **Graphs**, aquí en el menú desplegable seleccionamos la que queramos mostrar

30. SI hemos seguido estos pasos es más que probable que nuestras gráficas no estén funcionando, como dije antes, tenemos al SELinux actuando, basicamente porque no se consigue activar el agente, esto se puede ver en los archivos `log` tanto de servidor como de agente. Podemos echarle un ojo respectivamente en Ubuntu `/var/log/zabbix/zabbix_server.log` y en CentOS, `/var/log/zabbix/zabbix_agent.log` Tras buscar información he decidido desactivar el SELinux modifciando su archivo de configuración alojado en `/etc/selinux/config`

con mi editor y modificando la linea `SELINUX=enforce` por `SELINUX=disabled` Hecho esto reinicio la máquina.

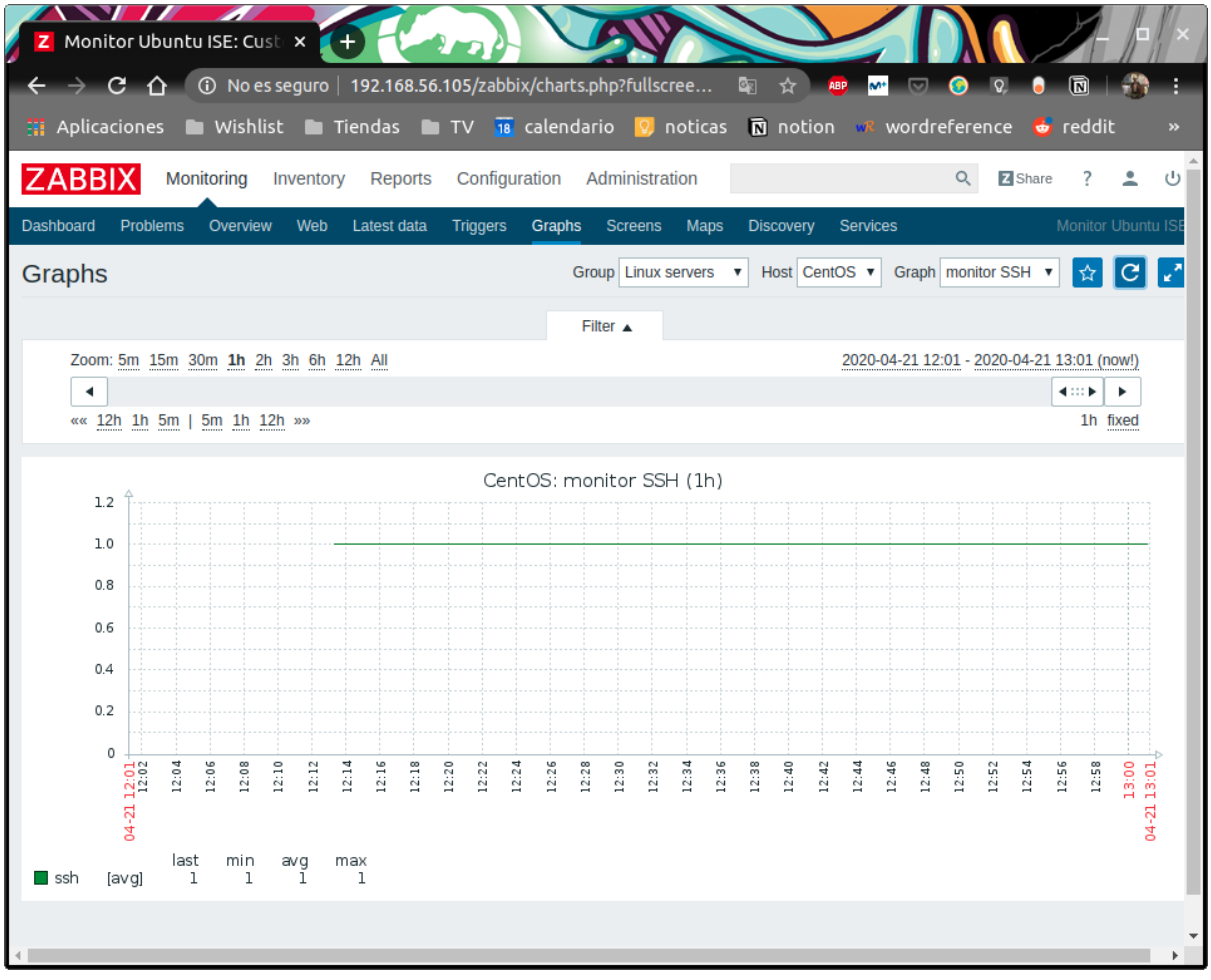
> Otro modo de arreglar esto es con `ausearch -c 'zabbix_agentd' --raw | audit2allow -M my-zabbixagentd`

31. En CentOS, accedemos a `/etc/zabbix/zabbix_agentd.conf` con nuestro editor favorito y modificamos las líneas `Server=ipdenuestroserver`, `ServerActive=ipdenuestroserver`, y `Hostanme=nombre que pusimos al crear el host en Zabbix`

32. Listo, reiniciamos sevicios en CentOS (`service zabbix-agent restart`) y en Ubuntu (`service zabbix-server restart` y `service apache2 restart`)

33. Debería de estar funcionando, si no es el caso mirar los `logs` y buscar información sobre posibles errores.

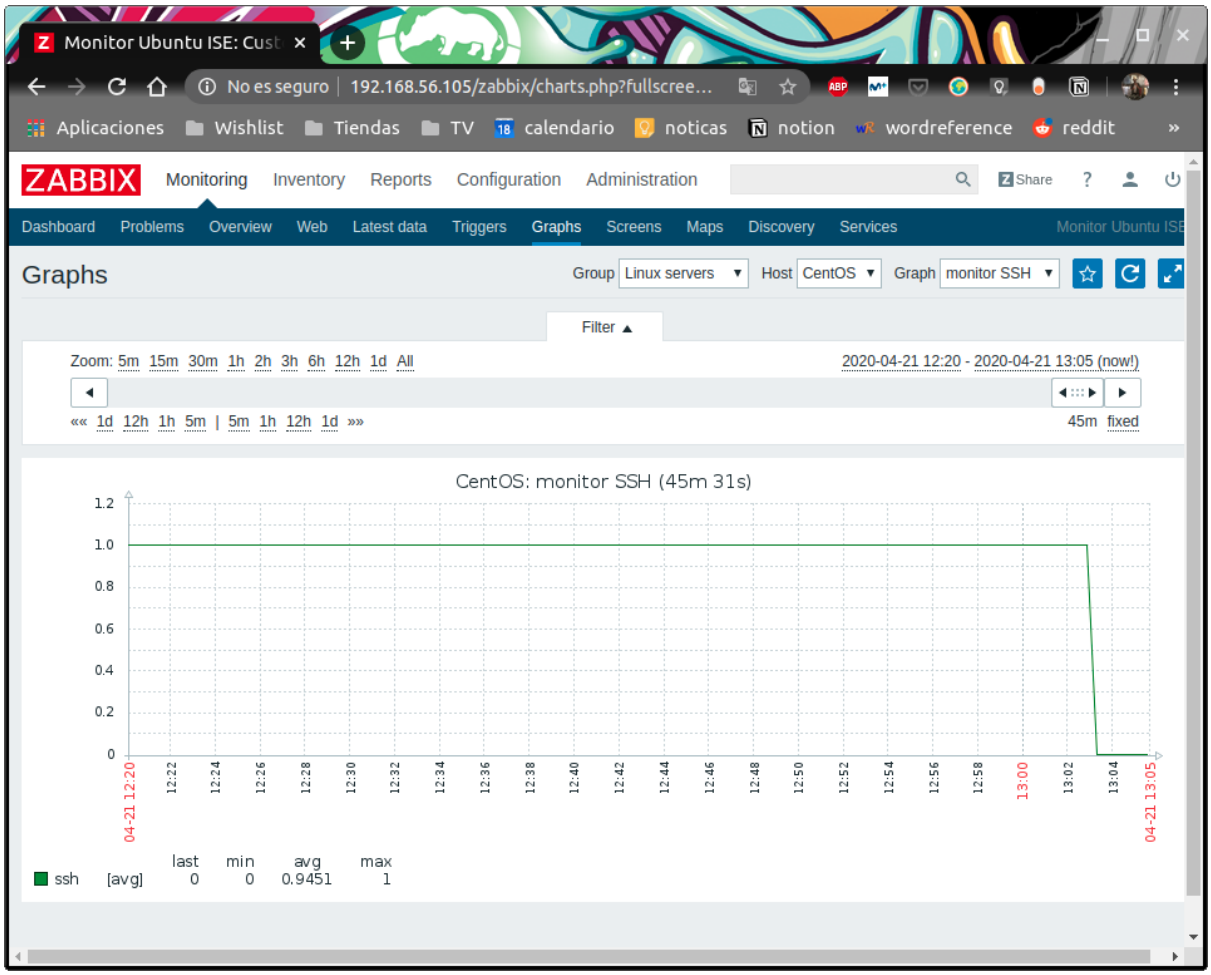
Oye, oye, muy bien eso de que esté funcionando, pero vamos a probarlooooo. Para ello, capturamos nuestra gráfica de `ssh` ahora que tenemos el servicio activo



Vemos que está a 1, significa que está activo, pero ¿y si desactivamos `ssh` en nuestra máquina CentOS?

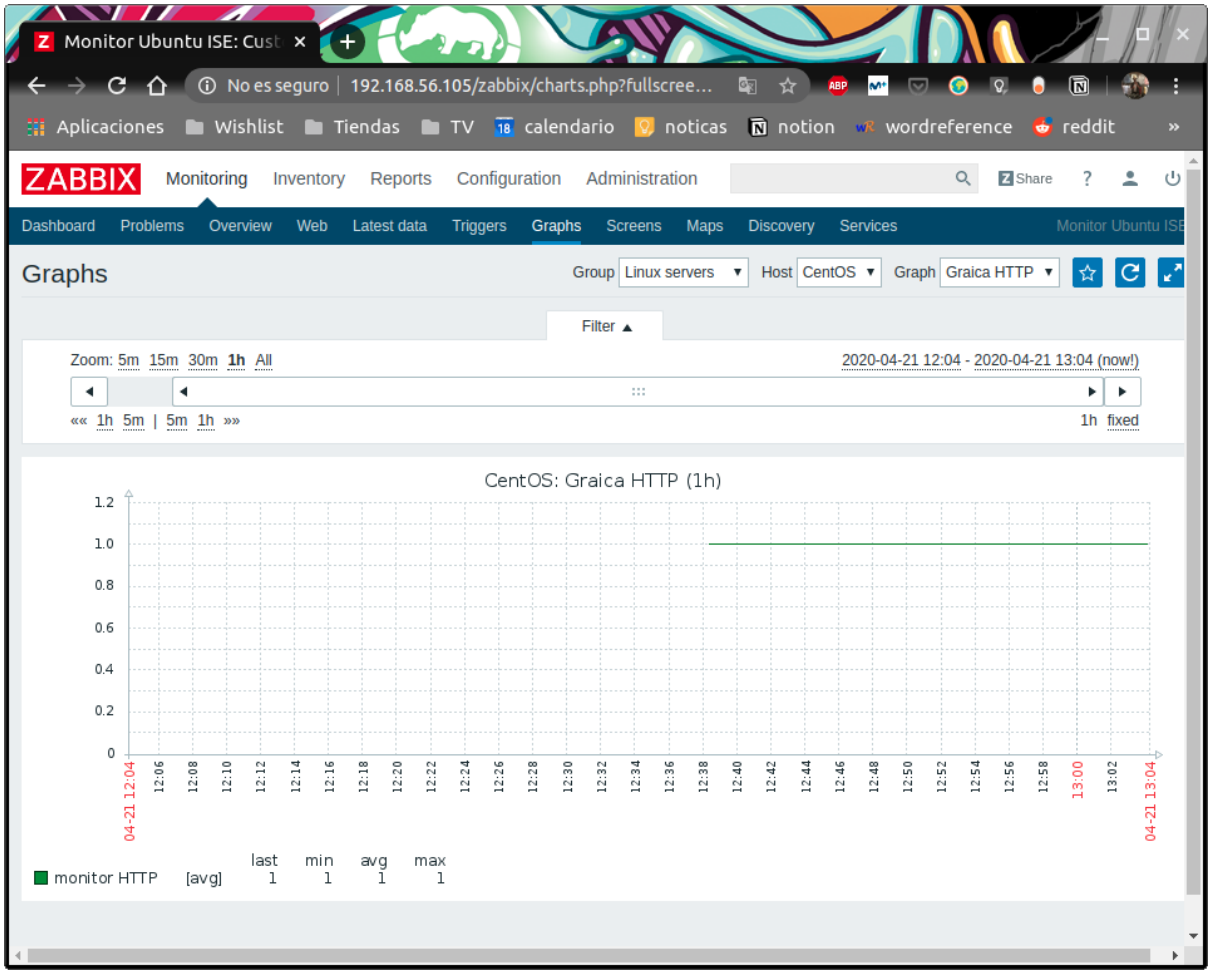
```
[root@localhost ~]# service sshd stop
Redirecting to /bin/systemctl stop  sshd.service
```

¿Que ha pasado en nuestra gráfica?




Pues como bien podemos observar su valor ha caído a 0, maravilloso. Esto demuestra que nuestro servicio `ssh` no escucha peticiones y por lo tanto, está desctivado.

¿Que tenemos en `http`?




Lo mismo, su puerto sigue escuchando, nos devuelve un valor 1, voy a descativarlo a ver que ocurre.

```
[root@localhost ~]# service httpd stop
Redirecting to /bin/systemctl stop  httpd.service
```


Pues sí, ha dejado de escuchar en su puerto 80, nuestro sistema de monitorización está funcionando.

Vamos finalmente a activar ambos servicios y ver como las gráficas se recuperan

```
[root@localhost ~]# service sshd start
Redirecting to /bin/systemctl start  sshd.service
[root@localhost ~]# service httpd start
Redirecting to /bin/systemctl start  httpd.service
```


Monitor Ubuntu ISE: Cust
No es seguro | 192.168.56.105/zabbix/charts.php?fullscree...
Aplicaciones
Wishlist
Tiendas
TV
calendario
noticas
notion
wordreference
reddit
ZABBIX
Monitoring
Inventory
Reports
Configuration
Administration
Share
Dashboard
Problems
Overview
Web
Latest data
Triggers
Graphs
Screens
Maps
Discovery
Services
Monitor Ubuntu ISE
Graphs
Group Linux servers
Host CentOS
Graph monitor SSH
Filter
Zoom: 5m 15m 30m 1h 2h 3h 6h 12h 1d All
2020-04-21 12:25 - 2020-04-21 13:10 (now!)
«« 1d 12h 1h 5m | 5m 1h 12h 1d »»
45m fixed
CentOS: monitor SSH (45m 31s)
1.2
1.0
0.8
0.6
0.4
0.2
0
12:25
12:26
12:28
12:30
12:32
12:34
12:36
12:38
12:40
12:42
12:44
12:46
12:48
12:50
12:52
12:54
12:56
12:58
13:00
13:02
13:04
13:06
13:08
13:10
04-21
04-21 13:10
last
min
avg
max
ssh
[avg]
1
0
0.8462
1


Todo perfecto, hasta aquí el paso a paso de como montar tu propio servidor de monitorización.