

# Manual de uso

ELKARBACKUP



## Aurkibidea

1.-¿ Qué es ElkarBackup ?.....	3
2.-Instalación.....	5
2.1.-Utilizar la imagen que hemos preparado.....	5
2.2.-Instalar el paquete en una Debian previamente instalada.....	6
2.3.-Actualizaciones.....	8
3.-Conectando con el servidor.....	9
4.-Clientes y Tareas.....	10
4.1.-Pero, ¿ Qué son los clientes y las tareas ?.....	10
4.2.-Añadiendo clientes GNU/Linux.....	11
4.3.-Añadiendo Clientes Windows .....	17
4.4.-Resolviendo problemas.....	23
5.-Políticas.....	25
Programación.....	26
Datos antiguos y desfasados.....	28
Uso de políticas distintas.....	28
6.-Scripts.....	30
6.1.-Un nuevo Script.....	30
7.-Configuración de la aplicación.....	33
7.1.-Gestionar parámetros.....	33
Claves SSH.....	33
Servidor MySQL.....	33
Mensajes a través de correo electrónico.....	33
Avisos de cuotas.....	34
Otros parámetros.....	35
7.2.-Gestionar el repositorio de los backups.....	36
7.3.-Copia del repositorio.....	37
Automatización.....	39
8.-Un repaso de conceptos: Rsnapshot.....	42
8.1.-Frecuencia.....	42
8.2.-Retención.....	43
8.3.-Rotación.....	43
9.-Anexos.....	45
9.1.-Tras descargara la imagen.....	45

---

9.2.-Script para comprimir el repositorio.....	45
9.3.-Ordenar tareas.....	47
9.4.-Transformando las imágenes a otros sistemas de virtualización.....	48
9.5.-Sincronización con la nube.....	48
9.6.-Copias y Snapshots de los clientes Windows.....	49
9.7.-Disco NFS remoto.....	49
10.-Licencia.....	50

## 1.- ¿ Qué es ElkarBackup ?

No es nuestra intención empezar a explicar lo que es un sistema de copias de seguridad, ni subrayar la importancia de garantizar la seguridad de los datos, o cuan importante es tener bien automatizado el proceso de la copia de seguridad. Lo que al menos tendríamos que tener claro es:

- Que ese amigo nuestro llamado Murphy suele venir de visita cuando menos se le espera, y que conviene estar preparado.
- Que deberíamos tener siempre a mano copias realizadas en momentos distintos: copias diarias, semanales, mensuales, etc.
- Que deberíamos prever la situación de catástrofe (si, esa en la que se pierde todo) externalizando los datos de manera sistemática.

El sistema de copias de seguridad corporativo ***ElkarBackup*** va a ser una solución más al alcance del administrador de sistemas, y estas son algunas de sus características:

- Se gestiona a través de un interfaz Web
- Realiza las copias a disco, no a cintas, por lo que será más rápido y no nos creará dependencias con dispositivos ***especiales (hardware de gestión de cintas)***.
- Utiliza enlaces duros (en adelante HardLinks<sup>1</sup>), por lo que el espacio que utiliza en disco estará muy bien optimizado.
- El interfaz está localizado en euskara, castellano e inglés.
- Hemos tratado que el interfaz sea auto-documentado para en la medida de lo posible evitar la necesidad de tener que acudir al manual.

Es una herramienta de software libre con licencia [GPL V3](#)<sup>2</sup>, y se basa en otras cuantas herramientas de software libre, sobre todo en rsnapshot<sup>3</sup> y rsync<sup>4</sup>. Aunque corre sobre sistemas GNU/Linux (hoy en día soportado sobre Debian), puede realizar copias de cualquier sistema operativo que soporte comunicación ssh/rsync.

Este proyecto se ha desarrollado como fruto de la colaboración entre el [IMH](#) y [Tknika](#) durante el curso 2012-2013, con el objetivo de dotar a los centros formativos y en general a cualquier organización de una herramienta para realizar y gestionar sus copias de seguridad de forma centralizada, superando las carencias que habíamos detectado en

---

1 [http://es.wikipedia.org/wiki/Enlace\\_duro](http://es.wikipedia.org/wiki/Enlace_duro)

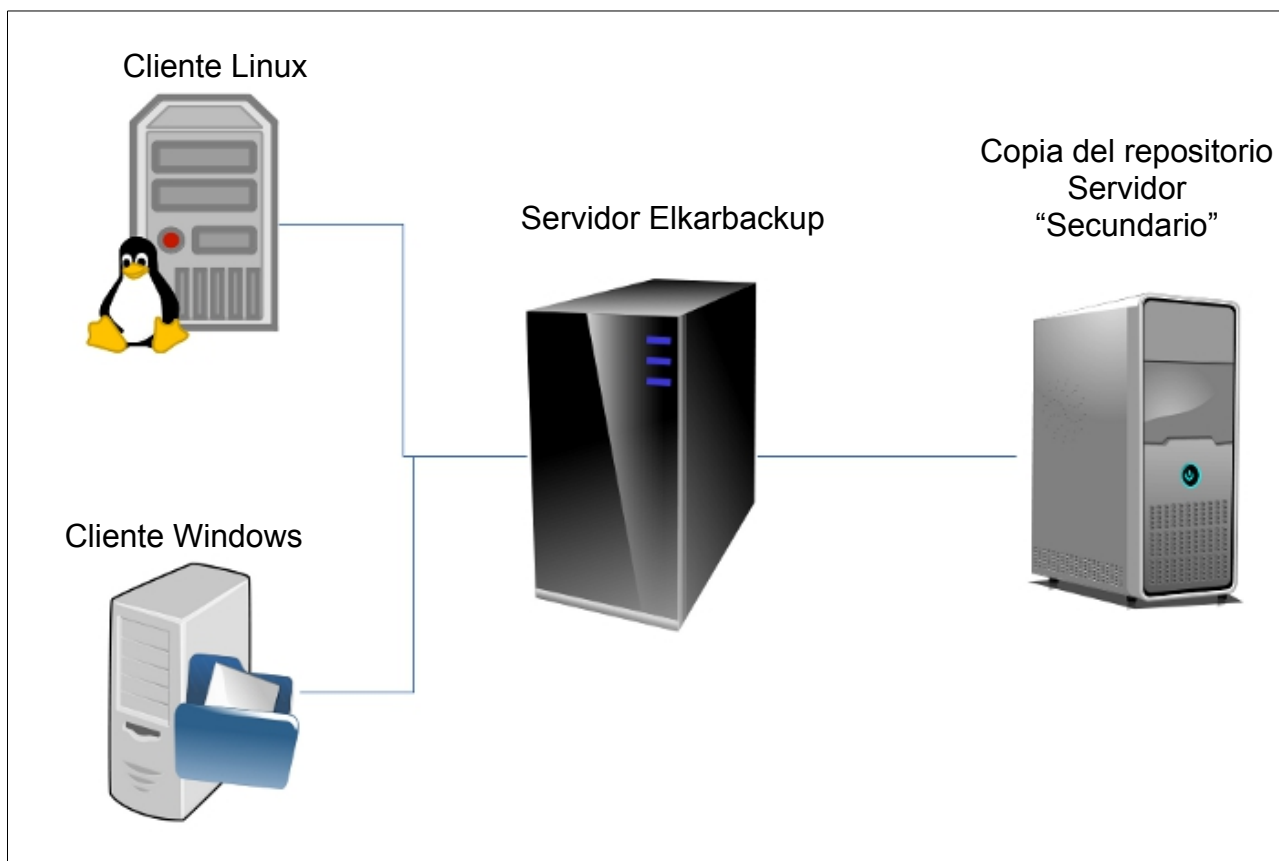
2 [http://en.wikipedia.org/wiki/GNU\\_General\\_Public\\_License#Version\\_3](http://en.wikipedia.org/wiki/GNU_General_Public_License#Version_3)

3 <http://www.rsnapshot.org/>

4 <http://es.wikipedia.org/wiki/Rsync>

otras herramientas similares. El desarrollo ha sido realizado por la empresa [Binovo](#), y el grupo [Elkarnet](#) ha colaborado en labores de testeo.

Durante el proceso que se describe en este manual veremos como se instala el servidor ElkarBackup, como lograr que este realice copias de ficheros de un equipo GNU/Linux y de un equipo Windows, y por último como realizar una copia de todo el repositorio en otro servidor.



## 2.- Instalación

Hay dos formas de poner en marcha la herramienta, una sería la de descargar y poner en producción en un sistema de virtualización una imagen que nosotros hemos preparado previamente, y la otra sería la de instalar el paquete ***elkarbackup*** en un servidor Debian previamente instalado.

### 2.1.- Utilizar la imagen que hemos preparado

Habría que descargarla imagen y añadirla al sistema de virtualización, que puede ser de distintas marcas. Nosotros recomendamos utilizar Proxmox y/o KVM, pero hay muchos más, como XEN, VMWARE, etc. En el apartado ***“Transformando las imágenes a otros sistemas de virtualización”*** que se encuentra al final de este documento, explicamos como se puede convertir una imagen RAW a los sistemas de virtualización VMWare y VirtualBox.

Las últimas imágenes se pueden descargar desde aquí:

- De 64 bits: <ftp://ftp.binovo.es/tknika-backups/images/>
- De 32 bits:

Como la imagen ya tiene toda la instalación hecha no haría falta mucho más para ponerla a trabajar, aunque hay un par de detalles que habrá que cambiar para adaptarla a nuestra red.

El servidor coge la dirección IP por DHCP, y esto habría que modificarlo ya que es conveniente que tenga una IP fija, para lo cual editamos el fichero ***/etc/network/interfaces***

```
root@ElkarBackup:~# nano /etc/network/interfaces
```

Y ahí le ponemos la IP, máscara, gateway y DNS.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
allow-hotplug eth0
# iface eth0 inet dhcp
iface eth0 inet static
address MI-IP
```

```
netmask MASCARA
gateway GATEWAY
dns-nameservers DNS
```

Probablemente cuando arranque no se activará el interfaz de red ya que se habrá dado cuenta de que su tarjeta (en este caso su tarjeta virtual) ha cambiado. Para solucionarlo editamos el fichero **70-persistent-net** borrando su contenido. Una vez que reiniciemos el servidor virtual el propio sistema se encargará de añadir en el fichero la información adecuada

```
root@ElkarBackup:~# nano /etc/udev/rules.d/70-persistent-net.rules
```

En la máquina virtual el usuario **root** tiene la contraseña **root**. Esto es algo que habría que cambiar cuando se ponga el servidor en producción.

```
root@ElkarBackup:~# passwd root
Introduzca la nueva contraseña de UNIX:
Vuelva a escribir la nueva contraseña de UNIX:
passwd: contraseña actualizada correctamente
```

Una vez realizados estos cambios habría que reiniciar la máquina virtual.

```
root@ElkarBackup:~# shutdown -r now
```

## 2.2.- Instalar el paquete en una Debian previamente instalada

Si en lugar de utilizar la imagen preferimos instalar el software en un servidor que tengamos con Debian, tendremos que realizar también la instalación de algunos otros paquetes de software. ElkaBackup utiliza una base de datos MySQL, por lo que en este punto habrá que tomar una decisión:

- O bien instalamos previamente en nuestro servidor ElkarBackup-Debian el paquete **mysql-server** para que el mismo sea quien gestione la base de datos
- O bien utilizamos otro servidor MySQL para gestionar la base de datos

El servidor MySQL se utilizará para guardar la configuración del sistema de backup: información de clientes y tareas, políticas, logs, usuarios, etc.

Además será necesario instalar otros paquetes: cliente mysql, PHP, etc. En nuestro caso los instalamos todos aquí (incluido el servidor mysql):

```
root@ElkarBackup:~# aptitude install debconf php5 php5-cli rsnapshot mysql-server php5-mysql acl bzip2
```

Al instalar el servidor MySQL nos pedirá que introduzcamos la contraseña de su usuario **root**. En nuestro ejemplo le hemos ponemos **root**, está claro que en producción habría que utilizar otra contraseña.

Ahora realizaremos la instalación del paquete ElkarBackup. Entre las dependencias que tiene destacaríamos:

- Servidor Web Apache: Lo utilizaremos para gestionar el interfaz web de los usuarios.
- Rsnapshot: El encargado de realizar las copias a disco utilizando HardLinks
- ssh y rsync: Software de comunicación y sincronización de datos con los clientes.
- Tiene que tener las ACL activadas en el sistema de ficheros.

Este punto es importante: El sistema tiene que tener instaladas y activadas las ACL en la partición que utilizar para guardar las copias. En Debian7 esto viene de serie, es decir, ya trae las ACL activadas en el sistema de ficheros, pero en Debian6 hay que activarlo de forma manual.

Si estamos instalando elkarbackup en una Debian6, tenemos que activar las ACL en **la partición root (/)**. Para ello editamos el fichero **/etc/fstab** y añadimos la palabra **acl**, tal y como se muestra a continuación

```
# / was on /dev/sda1 during installation
UUID=e3b77e85-df06-4659-b143-5939ccbf7d52 / ext3 errors=remount-ro,acl 0 1
```

Ahora lo mejor sería reiniciar el servidor para asegurarnos de que todo va bien.

Una vez que hemos arreglado el tema de las ACL, lo primero que haremos será importar la clave del repositorio. Lo haremos como usuario root:

```
root@backups:~# wget -O - ftp://ftp.binovo.es/tknika-backups/packages/repository_key.pub | apt-key add -
```

Editamos el **sources.list**

```
root@backups:~# nano /etc/apt/sources.list
```

Y añadimos estas líneas

```
# Binovo repository
deb ftp://ftp.binovo.es/tknika-backups/packages squeeze main
```

Por último actualizamos e instalamos:

```
root@ElkarBackup:~# aptitude update
root@ElkarBackup:~# aptitude safe-upgrade -y
root@ElkarBackup:~# aptitude install autofs elkarbackup
```

En la instalación necesita crear una base de datos MySQL para la aplicación, por lo que nos pedirá el nombre de usuario y contraseña del usuario administrador. Estos datos son



los que acabamos de introducir en la instalación del servidor MySQL, y decíamos que en nuestro ejemplo eran el usuario **root** con la contraseña **root**.

Por último otro detalle. Cuando nos conectamos a una máquina por SSH la primera vez, nuestro servidor nos dice que no conoce a esa máquina y que debemos confirmar su **fingerprint**<sup>5</sup>. Esta es una medida de seguridad, pero en nuestro caso y de cara a la automatización se convierte en problema, por lo que editaremos el fichero **/etc/ssh/ssh\_config** del servidor ElkarBackup y le añadiremos esto:

```
StrictHostKeyChecking no
```

Y de esta forma no pedirá la confirmación cuando nos conectemos a un nuevo servidor.

Con esto ya tenemos la aplicación instalada y lista para usar.

### 2.3.- Actualizaciones.

Sea cual sea el procedimiento de instalación elegido, la actualización de paquetes se realiza de la misma forma que el resto de paquetes en Debian:

```
root@backups:~# apt-get update  
root@backups:~# apt-get upgrade
```

---

5 Más información aquí:: <http://linuxcommando.blogspot.com.es/2008/10/how-to-disable-ssh-host-key-checking.html>

---

### 3.- Conectando con el servidor

Ahora con abrir un navegador web y poner la dirección del servidor **ElkarBackup** es suficiente para conectarnos a la aplicación. Tenemos dos opciones:

- Utilizar el nombre de **Host**: Como en la configuración de Apache **elkarbackup** es el nombre utilizado en la sección **ServerName**, podemos utilizar la dirección <http://elkarbackup> para lograr el acceso, siempre que en el DNS de la red se resuelva el nombre **elkarbackup** con la dirección IP del servidor.
- Utilizar la dirección IP: En este caso tendríamos que componer la dirección del modo siguiente: <http://DIRECCIONIP/elkarbackup/app.php/login>

Y nos validamos en la aplicación con el usuario **root** el cual tiene como password **root** (se puede cambiar dentro de la aplicación).

## 4.- Clientes y Tareas

La primera pantalla que vamos a ver es la de los **Clientes** y **Tareas**. Ahora se ve vacía pero pronto la iremos llenando.



Id	Nombre	Espacio en disco	Último evento	Estado	Acciones
----	--------	------------------	---------------	--------	----------

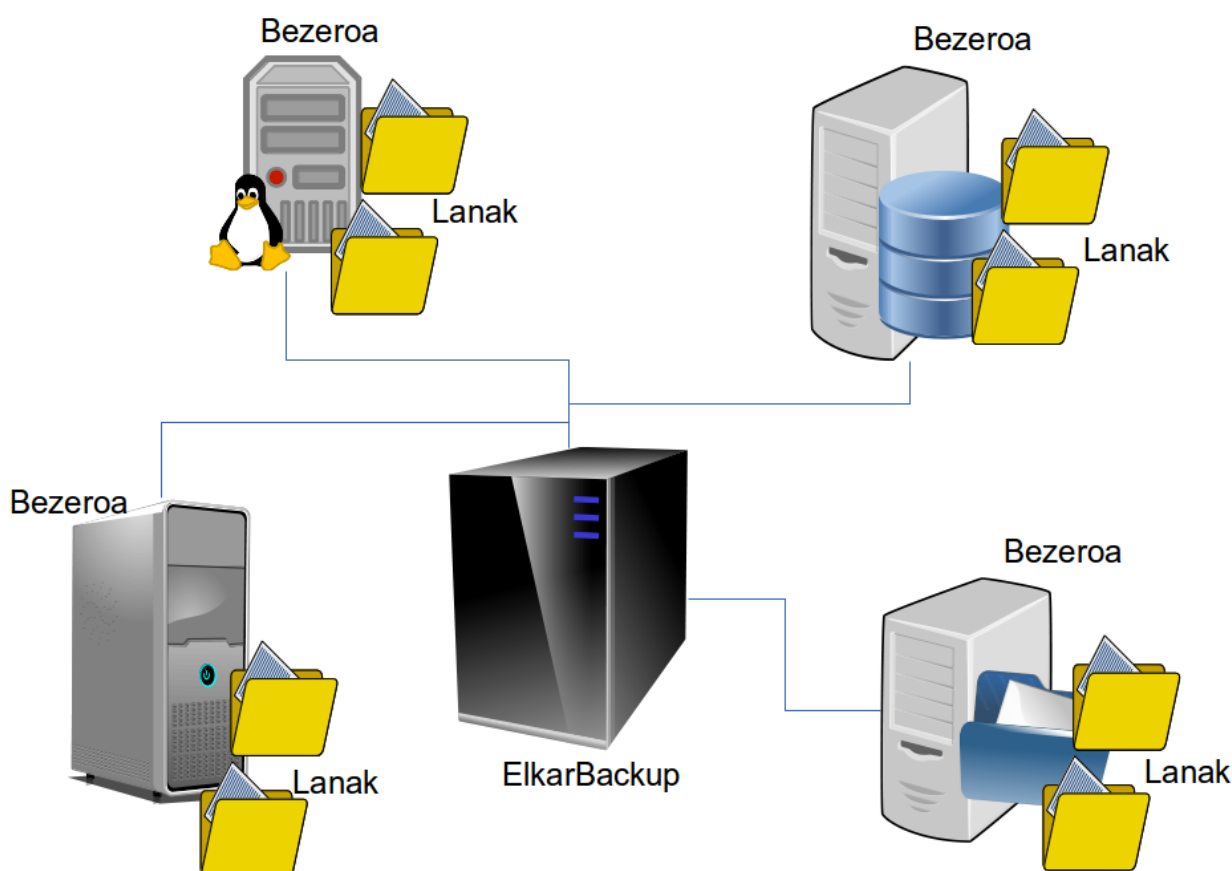
Nuevo

### 4.1.- Pero, ¿ Qué son los clientes y las tareas ?

En esta arquitectura hablaremos de **clientes** y **tareas** :

- **Cliente**: Una máquina GNU/Linux o Windows. Queremos copiar distintas carpetas que tenemos en esta máquina (que generalmente será un servidor).
- **Tarea**: Cada carpeta (raíz) que queremos copiar de cada cliente. A cada tarea se le podrá asignar una política distinta aunque sean del mismo cliente, cada una con su frecuencia y política de retención concreta.

Por ejemplo, imaginémonos que de un servidor queremos copiar por un lado los datos de los usuarios, y por otro algunos ficheros de log que genera un servicio concreto. Querremos copiar todos estos datos, pero con una programación y una política de retención distinta en cada caso.



Nuestro servidor podrá recoger y copiar información de múltiples clientes. La única condición que tiene que cumplir es la de soportar los protocolos **rsync** o **ssh**. En los servidores GNU/Linux no vamos a tener problemas, y para los servidores Windows instalaremos el software **Cwrsync**<sup>6</sup> que nos habilitará el servicio rsync (más adelante mostramos como se configura).

El uso de rsync en la comunicación tiene una ventaja importante: Antes de comenzar con la transferencia de ficheros ambos servidores deciden si hay necesidad de copiar el fichero, y solo se envían los ficheros modificados. Tendrá más carga de CPU el servidor que envía los ficheros y más E/S el que los recibe.

#### 4.2.- Añadiendo clientes GNU/Linux

Para añadir un cliente hay que hacer clic sobre el botón **Nuevo** y se nos mostrará un nuevo formulario para introducir los datos. Hemos intentado que la aplicación sea auto-

6 <https://www.itefix.no/i2/content/cwrsync-free-edition>

documentada, por lo que las explicaciones necesarias se encuentran en el propio formulario, y no vamos a repetirlas aquí.

Estos son los datos que utilizamos en nuestro ejemplo para el cliente GNU/Linux:

- Nombre: Cliente Debian
- URL: root@10.15.181.155
- CUOTA: -1 (Por ahora no la utilizaremos)
- Descripción: Un servidor Linux de nuestra red
- Pre/Post script: No seleccionamos nada

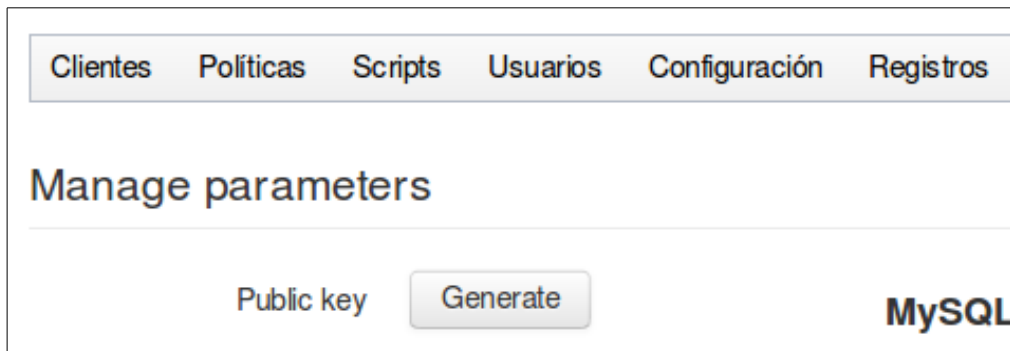
Si escogemos la opción del menú **Clientes → Mostrar** podremos observar que el nuevo cliente aparece en la lista, a pesar de que todavía no tiene tareas asignadas.

Cientes	Políticas	Scripts	Usuarios	Configuración	Registros	Sesión
<b>Cientes</b>						
Id	Nombre	Espacio en disco	Último evento	Estado	Acciones	
1	Cliente Debian	0 MB		Activo	Borrar	
						Nuevo

Pero teniendo en cuenta que en el proceso de backup la comunicación debe estar automatizada, ¿ cómo se conectara el servidor ElkarBackup con el nuevo cliente si en ningún sitio hemos puesto su contraseña?

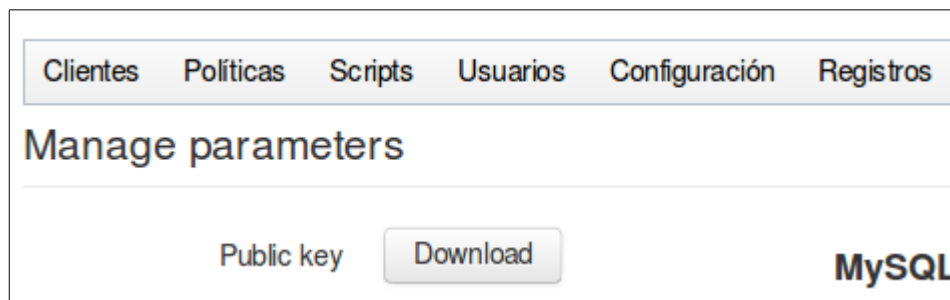
Como vamos a utilizar el protocolo ssh, lo haremos mediante claves públicas/privadas.

Tendremos que hacer clic una vez en el botón **Generar** que encontramos en el menú principal: **Configuración → Gestionar Parámetros** para que el servidor genere su par de claves.



Una vez generados el botón cambia y nos da la opción de descargar la clave pública.

En el menú principal: **Configuración** → **Gestionar Parámetros** → **Clave Pública: Descargar**



Si hacemos clic en el botón **Descargar** obtendremos en el fichero **Publickey.pub** la clave pública del servidor Elkarbackup. Debemos instalar esta clave pública en nuestros clientes GNU/Linux para que el servidor ElkarBackup pueda lanzar las conexiones de forma automática cuando llegue la hora de realizar las copias.

En las conexiones SSH que se crean para copiar los datos, los roles se reparten de la siguiente forma:

- Servidor ElkarBackup: Será el **Cliente** que abre la conexión.
- Cliente GNU/Linux: Será el **servidor** de la conexión ssh, por lo tanto tendrá que tener instalado el paquete **openssh-server** (en la mayor parte de los casos lo tendrá instalado).

Ahora veremos como instalar la clave **Publickey.pub** en el Cliente. En mi equipo personal lo descargo a través del navegador, y utilizando el comando **ssh-copy-id** lo instalo en el usuario **root** del cliente. No hay ni que decir que para poder hacer esto necesitamos conocer la contraseña del usuario root del cliente.

```
pedro@portatil59:~/Deskargak$ ssh-copy-id -i Publickey.pub root@10.15.181.155
root@10.15.181.155's password:
Now try logging into the machine, with "ssh 'root@10.15.181.155'", and check in:
```


```
~/.ssh/authorized_keys
```

to make sure we haven't added extra keys that you weren't expecting.

Con lo que hemos hecho hemos introducido en la configuración **ssh** del usuario **root** del cliente la clave pública generada en el servidor ElkarBackup, lo cual nos habilitará para lanzar conexiones automatizadas entre ambos en adelante.

```
root@DebianCliente:~# ls -la /root/.ssh/authorized_keys
-rw----- 1 root root 428 jun  7 11:11 /root/.ssh/authorized_keys
root@DebianCliente:~# cat /root/.ssh/authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQAC3H03zn8XhBW1JqxASqKEQZe+/fPcC47pu5l9c+s1Q+ppUD5C
LaDQjUsF+0VBHtPP2Wx1HaDidFURwge5GcnRnoXV32B+Vj9rfW9wPdk6siuZ2McoS0xyqbToS2CTdbS
xyjibn2CdM3RZjJa81ha0llciu38V9F1t4mylJVMxBgQmNwkTBwYGt66+wpT/nQVxiDeyVK98SHb8oG
KJZljpczlgYwckRAEPSARvxFm9yyW0ad3Qm7QYYGslBij4LXG1aUAARivoJdYtB4kb0nEd003n5BU/
0Q8eIqxDw7eFdnx4UDINX+mtSuHfpRko0RoU9IZZWGz9vrLnaxqh91G3
Automatically
generated key for tnikabackups.
```

Ahora vamos a crear una nueva tarea para copiar la carpeta **/media/Backups** del Cliente Debian, para lo cual editaremos la configuración del cliente y haremos clic en el botón **Agregar tarea** que se encuentra en la parte inferior



El formulario que se nos muestra también está auto-documentado y se explica la información que hay que introducir. Para el ejemplo introduciremos los siguientes datos, dejando el resto tal y como está:

- Nombre: Carpeta Backups
- Path: /media/Backups/
- Descripción: Datos que tenemos en la carpeta Backups
- Política: Default policy

Y haremos clic en botón **Guardar** para que se guarden los cambios. Si ahora volvemos a la vista general de los clientes veremos que la tarea ya se muestra.

Cientes	Políticas	Scripts	Usuarios	Configuración	Registros	Sesión
<b>Cientes</b>						
Id	Nombre	Espacio en disco	Último evento	Estado	Acciones	
1	Cliente Debian	0 MB		Activo	Borrar	
1.1	Cliente Debian/Carpeta Backups	0 MB		Activo	Borrar	Restore
						Nuevo

Dado que nunca se ha ejecutado, la columna **Último evento** todavía no muestra nada. En adelante veremos que en esta columna se mostrará el resultado y la hora de la última ejecución.

¿ Y cuándo tendremos los datos ? Pues esto dependerá de lo planificado en la **Política**. Nosotros le hemos asignado la política **Default policy**, y si miramos como está definida, podremos observar cual es su programación y su política de retención.

Para entender mejor el concepto de las políticas es conveniente tener claros los conceptos que se explican en el apartado **"Un repaso de conceptos: Rsnapshot"**, ya que es la lógica que se utiliza en nuestra aplicación. Al final profundizaremos más en esta parte.

Si dejamos que pase el tiempo para que se hagan las primeras copias y posteriormente hacemos clic sobre el botón **Restore** podremos observar el resultado



Bezeroak
Politikak
Script-ak
Erabiltzaileak
Konfigurazioa
Logak
Saioa

## Debian Bezeroa/Backups karpeta//

[Bezerora bueltatu](#)  
[Lanera bueltatu](#)

/var/spool/elkarbackup/backups/0001/0001

Izena	Aldaketa data	Ekintzak
.	2013-06-10 09:24:02	
..	2013-06-07 12:00:02	
.sync	2013-06-10 09:24:02	<a href="#">.tgz bezala deskargatu</a>
Hourly.0	2013-06-10 09:24:02	<a href="#">.tgz bezala deskargatu</a>
Hourly.1	2013-06-10 09:20:03	<a href="#">.tgz bezala deskargatu</a>
Hourly.2	2013-06-10 09:00:10	<a href="#">.tgz bezala deskargatu</a>

Podemos observar que la carpeta Hourly.0 contiene los datos de la última copia, y en las siguientes ejecuciones se irán acumulando más carpetas. Si nos vamos adentrando en cualquiera de ellas podremos llegar hasta el fichero concreto que nos interesa recuperar.

Bezeroak
Politikak
Script-ak
Erabiltzaileak
Konfigurazioa
Logak
Saioa

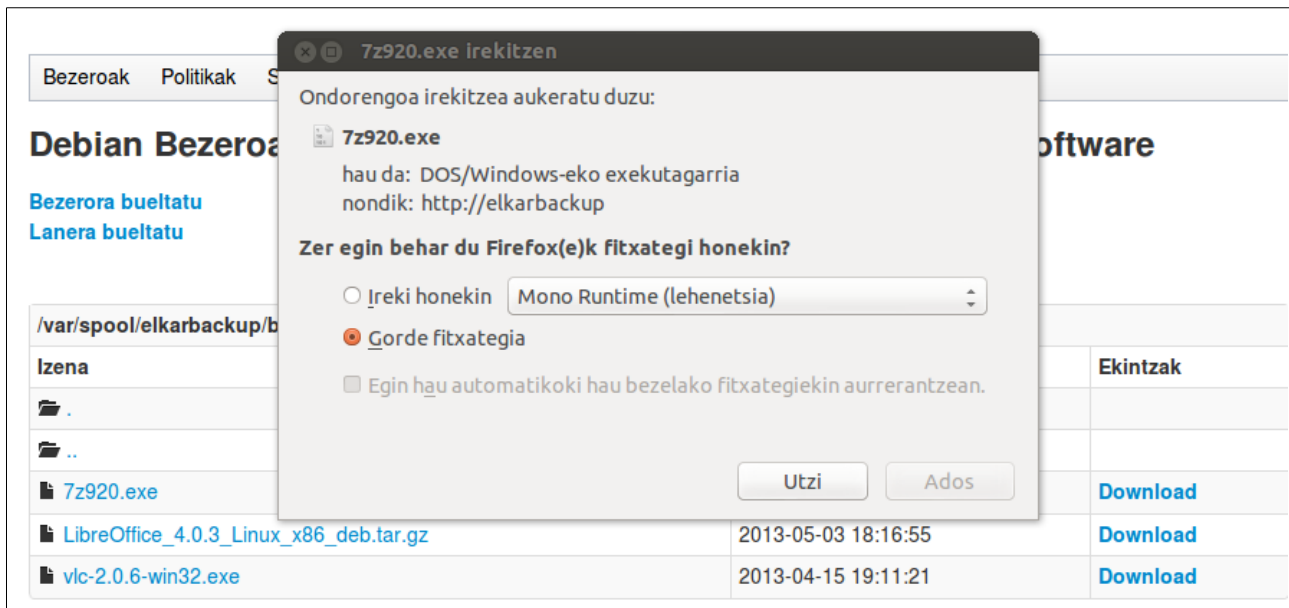
## Debian Bezeroa/Backups karpeta///Hourly.0/media/Backups/Software

[Bezerora bueltatu](#)  
[Lanera bueltatu](#)

/var/spool/elkarbackup/backups/0001/0001/Hourly.0/media/Backups/Software

Izena	Aldaketa data	Ekintzak
.	2013-06-07 11:24:19	
..	2013-06-07 11:24:47	
7z920.exe	2010-11-18 20:01:02	<a href="#">Download</a>
LibreOffice_4.0.3_Linux_x86_deb.tar.gz	2013-05-03 18:16:55	<a href="#">Download</a>
vlc-2.0.6-win32.exe	2013-04-15 19:11:21	<a href="#">Download</a>

Si queremos obtener el fichero, podemos descargarlo haciendo clic en el enlace



### 4.3.- Añadiendo Clientes Windows

La diferencia entre los clientes Windows y Linux está en el parámetro **URL**, el resto sería igual.

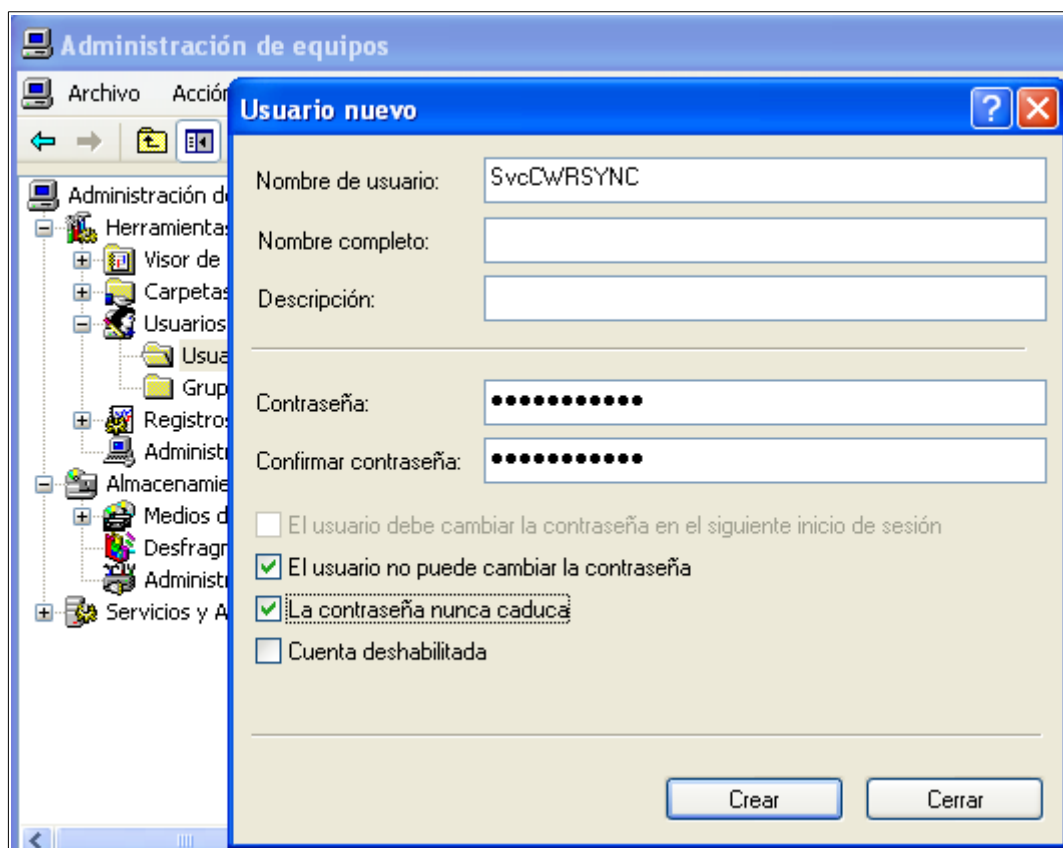
Con los servidores Windows utilizaremos el protocolo **rsync** en lugar de **ssh**. Para poder hacerlo, la máquina Windows tiene que aceptar conexiones rsync, es decir, necesita un servidor rsync activo.

- Nombre: Cliente Windows
- URL: **10.15.181.156:**
- CUOTA: -1
- Descripción: Un servidor Windows de nuestra red
- Pre/Post script: Lo dejamos sin seleccionar.

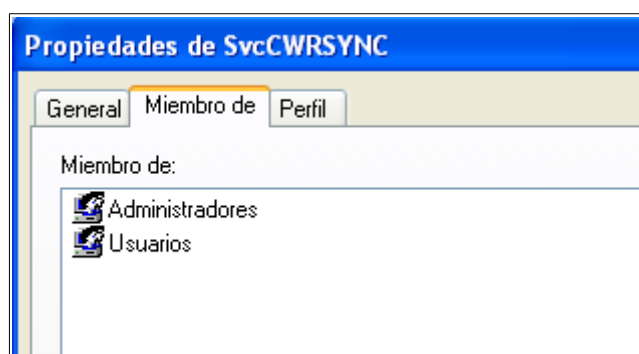
En el parámetro URL tras poner la IP tenemos que poner “:”, de esta forma le decimos que utilice el protocolo rsync.

Podemos encontrar distintos servidores Rsync para plataformas Windows. Nosotros utilizaremos la versión gratuita de [cwRsync](#), para lo cual descargaremos el programa **[cwRsyncServer 4.0.5 Installer](#)**.

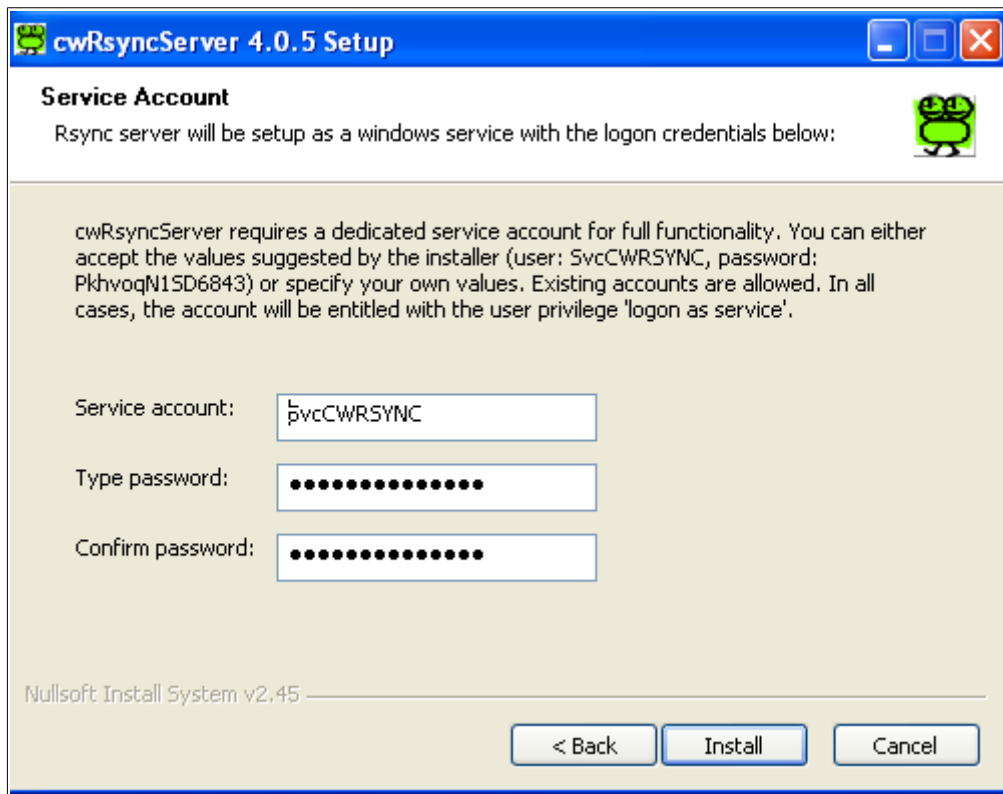
Antes de realizar la instalación añadiremos un usuario local en el servidor. En la máquina virtual con WindowsXP que utilizamos en el ejemplo hemos llamado al usuario **[SvcCWRSYNC](#)** y le hemos puesto como contraseña **[elkarbackup](#)**.



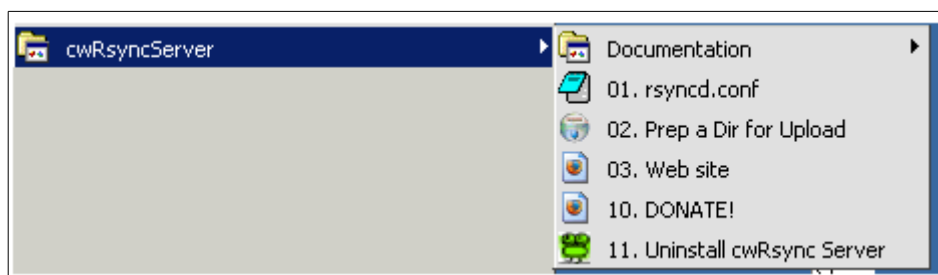
Dado que el servicio se va a ejecutar con este usuario, es conveniente que tenga los permisos necesarios, razón por la cual lo he añadido al grupo de administradores (probablemente con menos permisos también sería suficiente)



Ahora procedemos a la instalación del programa ***cwRsyncServer***. En la instalación nos pedirá el usuario y contraseña para este servicio, e introduciremos los que le acabamos de asignar:



Una vez finalizada la instalación nos aparecerá entre los programas disponibles, y editaremos el fichero ***rsyncd.conf*** para especificar cuales son las carpetas que queremos sincronizar



Esta sería la configuración para configurar la carpeta [C:\Backups](#)

Al principio ponemos estas dos líneas:

```
uid=0
gid=0
```

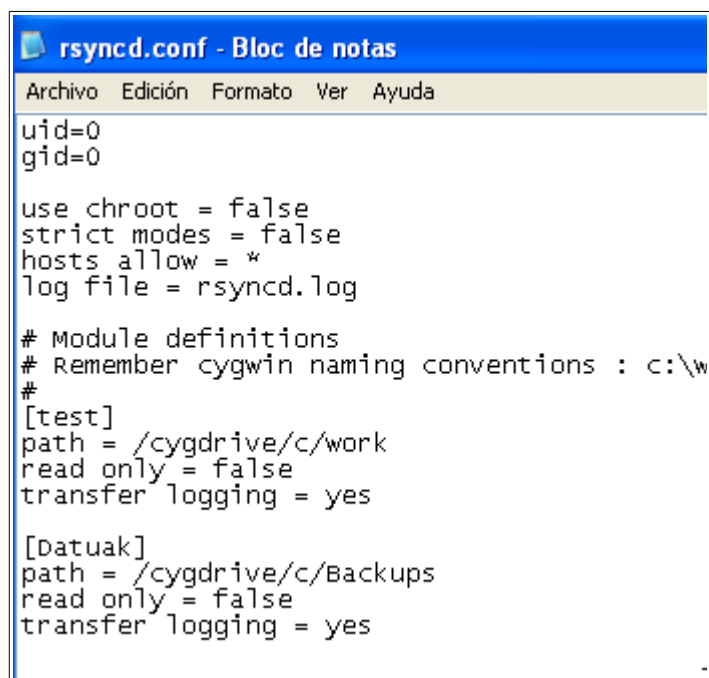
A continuación añadimos un bloque para cada carpeta que queremos sincronizar. En el ejemplo hemos nombrado la carpeta como ***[Datos]***:

```
[Datos]
path = /cygdrive/c/Backups
```

```
read only = false
transfer logging = yes
```

Si la carpeta que queremos copiar en vez de estar en [c:\Backups](#) estuviera en [d:\Backups](#), la línea path sería esta:

```
path = /cygdrive/d/Backups
```






```
rsyncd.conf - Bloc de notas
Archivo Edición Formato Ver Ayuda
uid=0
gid=0

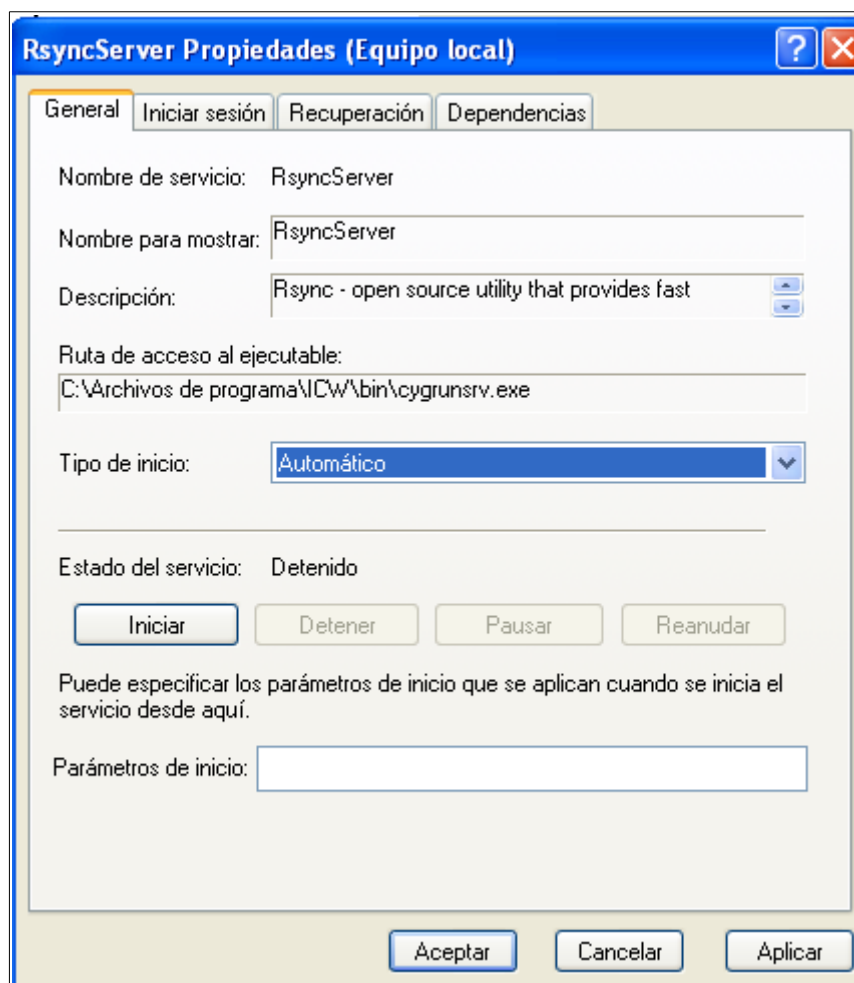
use chroot = false
strict modes = false
hosts allow = *
log file = rsyncd.log

# Module definitions
# Remember cygwin naming conventions : c:\w
#
[test]
path = /cygdrive/c/work
read only = false
transfer logging = yes

[datuak]
path = /cygdrive/c/Backups
read only = false
transfer logging = yes
```

Ahora comprobamos que el servicio está en marcha, y si no lo estuviera lo arrancaríamos especificando que el arranque tiene que ser automático

	Registros y alertas de rendimie...	Recopila información de ren...	Manual	Servicio de red
	RsyncServer	Rsync - open source utility t...	Iniciado	Automático
	Servicio COM de grabación de ...	Administra la grabación de ...	Manual	Sistema local



Ahora volvemos al interfaz de ElkarBackup y añadimos una tarea a nuestro cliente Windows para que haga una copia de su carpeta Backups.

Tenemos que tener en cuenta que esa carpeta en el fichero de configuración de la máquina Windows la hemos configurado en el bloque [**Datos**], y ese será el nombre que utilizaremos en el campo **Path**. Además le aplicaremos la política **Default policy**.

### Editar Tarea

Nombre Las copias de seguridad Client

Path Carpeta Backups

Espacio en disco 0 MB

Descripción La carpeta C:\Backups añadida en el bloque [Datos]

Está activo ☒

La guardaremos y haremos clic en el botón **Ejecutar ahora** para comprobar que la copia se hace bien. Esta opción nos permite lanzar la tarea sin esperar a que llegue ejecución programada.

La pantalla que nos da la visión general nos dará información también del estado de cada tarea, mostrando aquellas que están en espera (**QUEUED**) o en ejecución (**RUNNING**)

Bezeroak					
Id-a	Izena	Diskoaren erabilera	Azken log sarrera	Egoera	Ekintzak
1	Debian Bezeroa	187 MB	2013-06-07 16:41:03 OK	Aktibo	Ezabatu
1.1	Debian Bezeroa/Backups karpeta	187 MB	2013-06-07 16:41:03 OK	Aktibo	Ezabatu Restore
2	Windows Bezeroa	0 MB	2013-06-07 19:18:02 RUNNING	Aktibo	Ezabatu
2.2	Windows Bezeroa/Backups karpeta	0 MB	2013-06-07 19:18:02 RUNNING	Aktibo	Ezabatu Restore

Berria

Si esperamos un minuto veremos que la copia se ha realizado. En el *Log* también podemos ver información del resultado del proceso.

INFO	TickCommand		Command success: {"command":"elkarbackup:run_job","client":"2","job":"2"}	
INFO	StatusReport		OK	/client/2
INFO	RunJobCommand		Client "2", Job "2" du end.	/client/2/job/2
INFO	RunJobCommand		Client "2", Job "2" du begin.	/client/2/job/2
INFO	StatusReport		OK	/client/2/job/2
INFO	RunJobCommand		Client "2", Job "2" ok.	/client/2/job/2
INFO	RunJobCommand		Command "/usr/bin/rsnapshot" -c "/tmp/rsnapshot.2_2.cfg" Hourly 2>&1 succeeded with output:	/client/2/job/2
INFO	RunJobCommand		Running "/usr/bin/rsnapshot" -c "/tmp/rsnapshot.2_2.cfg" Hourly 2>&1	/client/2/job/2
INFO	RunJobCommand		Command "/usr/bin/rsnapshot" -c "/tmp/rsnapshot.2_2.cfg" sync 2>&1 succeeded with output:	/client/2/job/2
INFO	RunJobCommand		Running "/usr/bin/rsnapshot" -c "/tmp/rsnapshot.2_2.cfg" sync 2>&1	/client/2/job/2
INFO	StatusReport		RUNNING	/client/2/job/2

#### 4.4.- Resolviendo problemas

Para la transferencia de los ficheros nos estamos basando en las comunicaciones entre servidores, y estas a veces suelen ser problemáticas. El apartado de **Logs** puede ser muy útil al detectar la causa de estos problemas.

Al escribir este documento he tenido un problema de comunicación con el cliente Windows, y esto es lo que veía en el apartado de **Logs**:

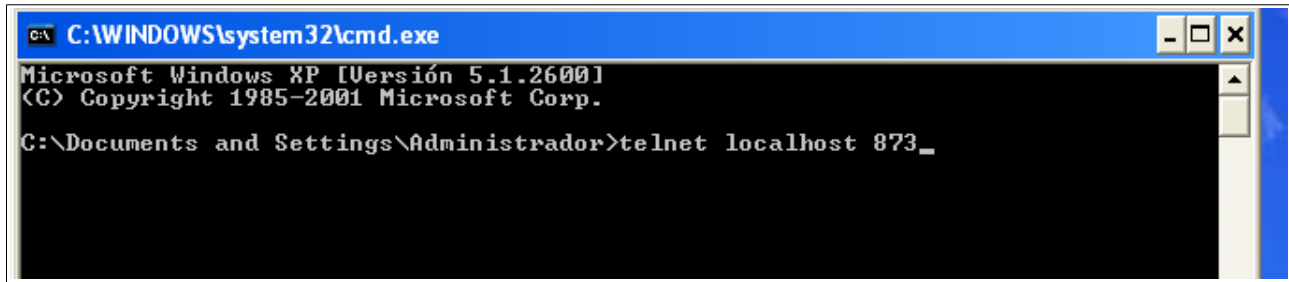
```
Command "/usr/bin/rsnapshot" -c "/tmp/rsnapshot.2_2.cfg" sync 2>&1 failed.
Diagnostic information follows: rsync: failed to connect to 10.15.181.156:
Connection timed out (110) rsync error: error in socket IO (code 10) at
clientserver.c(122) [Receiver=3.0.7]
```

El sistema me dice que tiene problemas de comunicación con el cliente a través del protocolo rsync. Puede haber al menos dos causas:

1. El servicio rsync del cliente Windows va mal.
2. Tenemos problemas de comunicación con el cliente y el protocolo Rsync

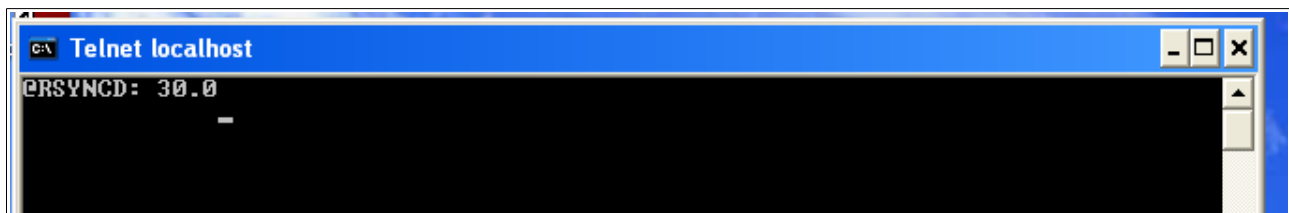
Para ver si es la primera opción hacemos un telnet al puerto rsync (TCP 873) desde el mismo cliente Windows.





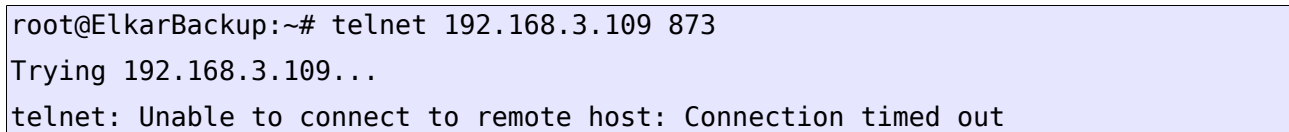
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Versión 5.1.2600]
<C> Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Administrador>telnet localhost 873_
```

Y si vemos que responde eso significa que el servicio está bien



```
Telnet localhost
ERSYNCD: 30.0
-
```

Intentamos la misma conexión pero esta vez desde el servidor ElkarBackup hasta el cliente Windows:



```
root@ElkarBackup:~# telnet 192.168.3.109 873
Trying 192.168.3.109...
telnet: Unable to connect to remote host: Connection timed out
```

vemos que la conexión no se establece, por lo que todo apunta a algún cortafuegos, seguramente en el cliente Windows.

Tras parar el servicio de cortafuegos en el cliente Windows se ha resuelto el problema. Esto no significa que haya que deshabilitar el cortafuegos, pero al menos habría que permitir el tráfico **rsync** (TCP 873) entre el servidor ElkarBackup y el cliente Windows.

## 5.- Políticas

Con lo visto hasta ahora hemos conseguido realizar copias, pero en un sistema de backups hay que definir más cosas:

- Programación: ¿ Cuándo se hacen las copias ?
- Retención: ¿ Hasta cuándo hay que mantener las copias antiguas ?
- ¿ Hay que mezclar distintas programaciones ? Copias durante el día, diarias, semanales, mensuales ....

Para definir todos estos conceptos utilizaremos **Políticas** (hasta ahora hemos utilizado la denominada **Default Policy**). Dado que ElkarBackup se basa en el software rsnapshot es conveniente repasar los conceptos que se explican en el apartado **"Un repaso de conceptos: Rsnapshot"**.

La aplicación nos dará la opción de definir distintas políticas. A cada tarea le asignaremos una política, y una política podrá ser reutilizada en más de una tarea.

Estos son los datos que tendremos que definir al añadir una nueva política:

- Nombre y Descripción: Campos de texto libre
- Excluir: Patrones para excluir ficheros de la copia
- Incluir: Patrones para incluir ficheros en la copia, a pesar de que estos hayan quedado excluidos por el patrón introducido en el campo **Excluir**. Por ejemplo, imaginémonos que en general no queremos incluir en la copia los ficheros de vídeo, por lo que ponemos \*.avi en el campo **Excluir**, pero si que nos interesa copiar los ficheros avi que se encuentran en la carpeta del director. En este caso en el campo **Incluir** añadiríamos **director/\*.avi** para esta excepción.
- Sincronizar primero: cuando rsnapshot realiza una nueva copia, realiza una rotación de las copias realizadas previamente y genera la nueva copia en una nueva carpeta (por ejemplo Daily.0), incorporando en la misma tanto los ficheros **que han sido modificados** como los ficheros **que no han sido modificados**, de modo que todos estén ubicados en una misma carpeta. Los ficheros que no habían sido modificados los enlaza mediante **hardlinks**, de forma que no ocupen más espacio físico en el disco.

Este es un proceso que consume tiempo, ya que al colocar cada fichero debe decidir si es una nueva copia o si tiene que enlazarlo con un hardlink, y de mientras el cliente remoto sigue atendiéndole. En algunos casos no tendrá mayor importancia, pero en otros casos será importante liberar al cliente cuanto antes. En este último caso podemos escoger la opción **Sincronizar primero**.

Cuando elegimos esta opción le decimos a rsnapshot que primero haga una sincronización de los ficheros en una carpeta llamada **.sync** para liberar cuanto antes al cliente. Cuando finalice esta parte y tras dejar libre al cliente, comparará los datos de la última copia (por ejemplo Daily.0) con los que hay en la carpeta **.sync**, y generará la nueva estructura añadiendo los hardlink necesarios.

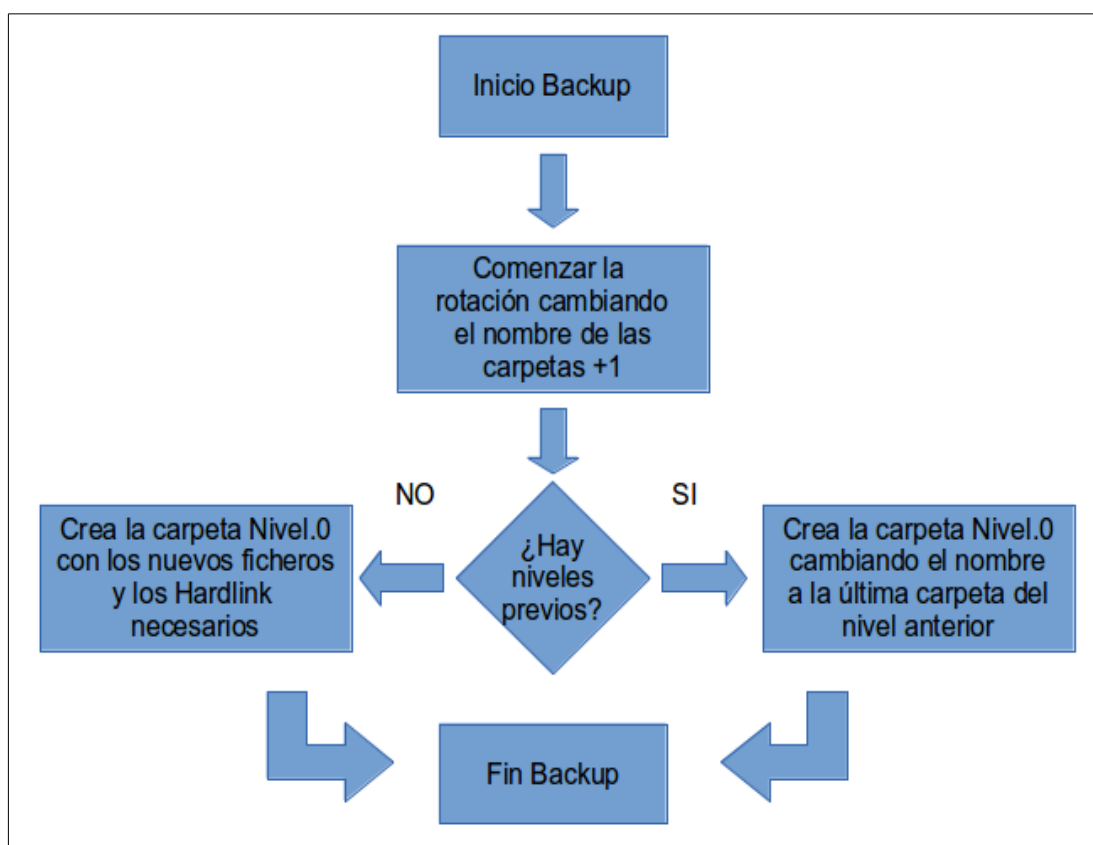
Por lo tanto, y respondiendo a la lógica que nos da a entender su nombre, cuando escogemos esta opción el sistema primero hará una sincronización de datos con el cliente y posteriormente creará la estructura de carpetas y hará las rotaciones pertinentes.

### Programación

Este es otro punto importante que hay que entender. Por un lado podemos ver el área de configuración de la programación compuesto por 5 fichas:

- Hourly o Durante el día: Copias que se realizarán durante el día en horas distintas.
  1. Primero hará la rotación de las carpetas Hourly, borrando la última y sumando +1 al resto.
  2. Crear y llenar la carpeta Hourly.0: Ficheros nuevos y los hardlink que estime necesarios
- Daily o Diarios: Copias a realizar una vez al día a una hora determinada y los días de la semana que se hayan seleccionado. Hay que tener en cuenta que si en esta política se han activado niveles anteriores aquí no se realizarán copias, sino rotaciones: El último Hourly pasará a ser el Daily.0.
  1. Primero hará la rotación de las carpetas Daily, borrando la última y sumando +1 al resto.
  2. Se utiliza el nivel anterior (Hourly) ?
    - Si: Se crea el Daily.0 renombrando la última carpeta Hourly
    - No: Crea y llena la carpeta Daily.0: Ficheros nuevos y los hardlink que estime necesarios
- Weekly o Semanal: Copia a realizar una vez a la semana en un día y hora determinados. Hay que tener en cuenta que si en esta política se han activado niveles anteriores aquí no se realizarán copias, sino rotaciones: El último Daily pasará a ser el Weekly.0.

1. Primero hará la rotación de las carpetas Weekly, borrando la última y sumando +1 al resto.
2. Se utiliza el nivel anterior (Daily) ?
  - Si: Se crea el Weekly.0 renombrando la última carpeta Daily
  - No: Crea y llena la carpeta Weekly.0: Ficheros nuevos y los hardlink que estime necesarios
- Monthly o Mensual: Se repite la lógica descrita en los puntos anteriores
- Yearly o Anual: Se repite la lógica descrita en los puntos anteriores



En más de una ocasión nos hemos referido a ***la última carpeta*** de cada nivel. Eso tiene mucho que ver con la política de retención. Cuando decimos que en un nivel determinado la retención es N, esto significa que las carpetas que se generarán en ese nivel se nombrarán desde 0 a (N-1).

Por ejemplo, si definimos que las copias a realizar durante el día tienen retención de 4, eso significa que el sistema guardará 4 carpetas: Hourly.0, Hourly.1, Hourly.2 y Hourly.3.

Cuando llegue el momento de realizar la rotación de las carpetas Hourly hará lo siguiente:

Borra el último Hourly:

```
rm -Rf Hourly.4
```

Cambia el nombre al resto para hacer la rotación

```
mv Hourly.3 Hourly.4  
mv Hourly.2 Hourly.3  
mv Hourly.1 Hourly.2  
mv Hourly.0 Hourly.1
```

y por último crea un nuevo Hourly.0 con su contenido, copiando los nuevos ficheros y enlazando mediante Hardlinks aquellos que no han sido modificados.

### Datos antiguos y desfasados

Imaginémonos que en una política tenemos puesta una retención de 6 para los Hourly, esto significa que tendremos las carpetas Hourly.0, ..., Hourly.5 con los datos de las últimas copias. Por alguna razón pensamos que es excesivo, y modificamos la retención a 4, por lo tanto utilizará las carpetas Hourly.0, ..., Hourly.3.

¿Y qué ocurre con las carpetas Hourly.4 y Hourly.5? Pues que ahí estarán hasta que nosotros las borremos. No se actualizarán y se quedarán desactualizadas porque tampoco entrarán en el ciclo de rotaciones, pero estarán ahí, ***"haciendo ruido"***. Lo mejor en estos casos es que borremos a mano esas carpetas.

¿Y no sería mejor que en estos casos el sistema eliminara estas carpetas de forma automática? Podría ser, pero el eliminar datos de forma automática también tiene sus riesgos. Imaginémonos que en una política que tenemos ***en producción*** estamos haciendo cambios y que por error modificamos el parámetro de retención poniendo un valor demasiado pequeño, y que en consecuencia el sistema elimina automáticamente las carpetas con las copias. En la medida de lo posible debemos evitar la pérdida de datos, por lo que es mejor borrar a mano estas carpetas cuando sea necesario.

### Uso de políticas distintas

En nuestras redes tenemos distinto tipo de información, y el tener que tener copias de seguridad de todos estos datos no significa que debemos darles a todos el mismo tratamiento.

---

En algunos casos la información deberá ser guardada por mucho tiempo (meses o años), y en otros casos tras una semana los datos no tendrán validez y quisiéramos borrarlos.

Por otro lado, para un tipo de datos será suficiente con hacer una copia a la noche, y en cambio para otros quisiéramos tener 3 copias programadas durante el día.

Podremos crear distintas políticas para distintas necesidades, y luego será suficiente aplicar la política adecuada a cada **tarea**.

Es importante comprender que las distintas tareas de un **Cliente** pueden estar programadas con distintas políticas, ya que **las políticas se enlazan con tareas**, no con clientes.

## 6.- Scripts

En muchos casos nos interesará hacer ***algo*** antes y/o después de la ejecución de una tarea, por ejemplo:

Antes:

- Tal vez necesitamos abrir una conexión VPN para conseguir conexión con un servidor remoto
- Crear un [snapshot](#)<sup>7</sup> de todo el disco de un cliente remoto para copiar los datos sin interferencias
- Detener un servicio para copiar sus datos (zimbra, MySQL, ....)
- etc.

Después:

- Cerrar la conexión VPN que habíamos abierto
- Asegurarnos de que ciertos datos son coherentes: fechas creación de ficheros, ficheros comprimidos bzip que se descomprimen sin errores, ...
- etc.

Utilizaremos los scripts para programar estas acciones, y tendremos autonomía para poder desarrollar nuevos scripts e incluso para compartírlas con otros.

### 6.1.- Un nuevo Script

Estos son los datos que nos pide al crear un nuevo script:

- Nombre y Descripción
- Fichero: Para cargar el script que previamente hemos programado en nuestro ordenador
- Opciones de ejecución: Dependiendo de la lógica de cada script, algunos estarán pensados para ser ejecutados ***a nivel de cliente***, mientras que otros pueden estar pensados para ejecutarse ***a nivel de tarea***. De ellos, algunos para se ejecutados como ***Pre-script*** (a realizar antes de que se ejecute la acción) y otros para ser ejecutados como ***Post-script*** (a realizar después de que se ejecute la acción). Nosotros decidiremos como pueden ser ejecutados, y en función de esa decisión tendremos la opción de escogerlos en los clientes y/o tareas.

---

<sup>7</sup> [http://es.wikipedia.org/wiki/Copia\\_instant%C3%A1nea\\_de\\_volumen](http://es.wikipedia.org/wiki/Copia_instant%C3%A1nea_de_volumen)

- Variables de entorno: Dado que los scripts pueden utilizarse en más de un sitio, al escribirlos desconocemos para qué cliente/carpeta se van a ejecutar, y en la mayoría de los casos esta información será necesaria. En nuestros scripts podremos utilizar las siguientes variables de entorno:
  - ELKARBACKUP\_LEVEL: Su valor puede ser JOB o CLIENT
  - ELKARBACKUP\_EVENT: Su valor puede ser PRE o POST
  - ELKARBACKUP\_URL: Su valor será la URL completa de la tarea o cliente
  - ELKARBACKUP\_ID: Código ID de la tarea o cliente (número)
  - ELKARBACKUP\_PATH: Path completo de la carpeta raíz
  - ELKARBACKUP\_STATUS: Estado de salida de Post-scripts, en principio siempre 0.

Si vamos a visualizar o editar la información de un script previamente creado, podremos ver dónde está siendo utilizado, y también tendremos la opción de descargarlo a nuestro equipo.

Los scripts se guardan en la carpeta ***/var/spool/ElkarBackup/uploads*** del servidor.



Cientes Políticas Scripts Usuarios Configuración Registros Sesión

## Edit script

Download

Nombre

Reducir el repositorio

Descripción

File

Browse

Runs as before client script

☐

Runs as after client script

☐

Runs as before job script

☐

Runs as after job script

☒

Currently not used

Guardar

Hemos puesto un ejemplo de Script en el apartado ***Script para comprimir el repositorio.***

## 7.- Configuración de la aplicación

Dentro del menú principal en la opción **Configuración** podemos configurar distintos parámetros de la aplicación.

### 7.1.- Gestionar parámetros

#### Claves SSH

Ya [hemos hablado algo](#) acerca de la clave pública. Desde aquí podemos descargar la clave pública del servidor ElkarBackup que previamente se ha generado desde este mismo sitio.

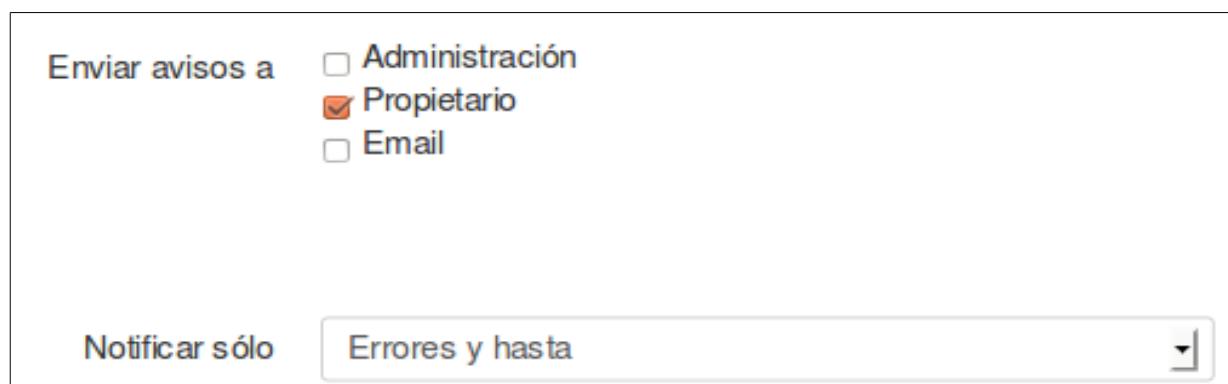
#### Servidor MySQL

A continuación tenemos los parámetros para gestionar la configuración del servidor MySQL. Los ficheros de backup que se copian con Rsnapshot se guardan en disco, pero el resto de los datos (datos de clientes y tareas, políticas, etc) se guardan en la base de datos.

El servidor MySQL puede estar ubicado en el servidor ElkarBackup o en cualquier otro servidor de nuestra red, aquí será donde se configurará la ubicación y los parámetros de conexión.

#### Mensajes a través de correo electrónico

Si nos interesa recibir alertas por correo, debemos decidir que nivel de alertas queremos que envíe y a quién a de enviárselas.



En la configuración predeterminada se enviarán las alertas al **propietario** (esto lo veremos más adelante), y los mensajes que enviará serán del tipo **Errores y superior**. Si

nos interesa que además de al propietario las alertas se envíen también a otra dirección de correo, tendríamos que seleccionar la opción **Email** y añadir la nueva dirección en la caja de texto que se muestra.



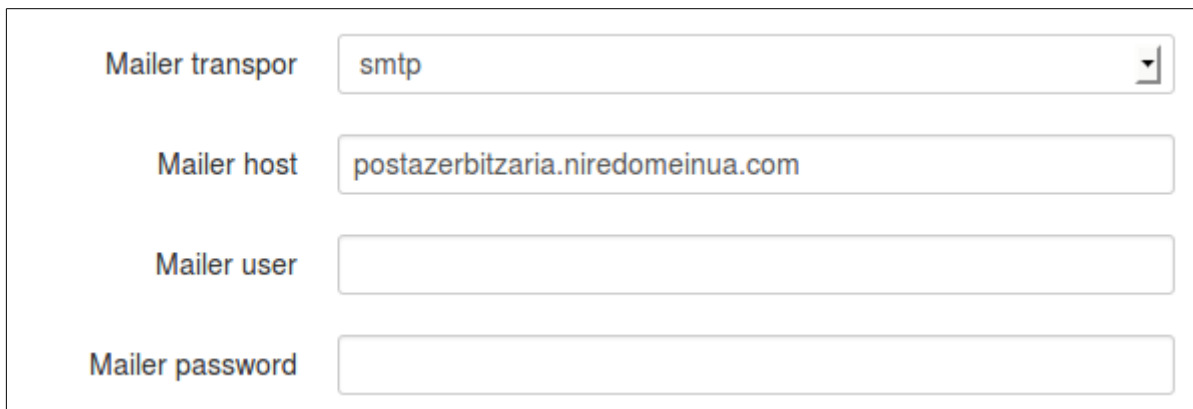
Enviar avisos a

- ☐ Administración
- ☒ Propietario
- ☒ Email

nombre@midominio.com

Notificar sólo Errores y hasta

Pero el servidor también necesita ser configurado para el envío de correos, es decir, necesita saber cómo tiene que enviar los mensajes, y para ello tenemos los parámetros **Mailer**.



Mailer transpor smtp

Mailer host postazerbitzaria.niredomeinua.com

Mailer user

Mailer password

En la programación de esta sección se han utilizado los recursos del framework Symfony, por lo que podemos encontrar más información al respecto en [la documentación del proyecto Symfony](#)<sup>8</sup>.

#### Avisos de cuotas

Hasta el momento no hemos hablado de las cuotas, pero podremos definir cuál va a ser la cuota de utilización de disco para cada cliente. Esto puede llegar a ser práctico cuando un cliente consume mucho espacio y no queremos perjudicar al resto de clientes que pueden quedarse sin espacio en disco.

<sup>8</sup> <http://symfony.com/doc/current/cookbook/email/email.html>

Si definimos un máximo en el parámetro **CUOTA** de un cliente y es sobrepasado, lo veremos resaltado tanto desde la vista de cliente como desde la vista principal de clientes y tareas.

Quota	512000
Espacio en disco	705 MB (141%)

Clientes					
Id	Nombre	Espacio en disco	Último evento	Estado	Acciones
1	Cluster	705 MB (141%)	2013-06-19 15:15:47 OK	Activo	<button>Borrar</button>
1.1	Cluster/Orokorra karpeta	705 MB	2013-06-19 15:15:46 OK	Activo	<button>Borrar</button> <button>Restore</button>

Hay que tener en cuenta que ***no se realizan copias de las tareas de un cliente que ha sobrepasado su CUOTA.***

Antes de que este límite sea sobrepasado el sistema nos enviará alertas. Mediante este parámetro definiremos el umbral a partir del cual se enviarán estas alertas, por defecto cuando el uso llegue al 80% de la cuota.

Quota warning level	80	%
---------------------	----	---

#### Otros parámetros

- Cuánto tiempo mantendrá la información de LOG antes de eliminarla
- Número máximo de líneas por página que mostrará
- Prefijo que utilizará el sistema para generar ciertas URLs si en vez de utilizar un nombre resuelto por DNS para acceder al interfaz (por ejemplo <http://elkarbackup>) se utiliza una dirección IP fija (por ejemplo <http://IP/elkarbackup/app.php/login>), (en el ejemplo **/elkarbackup**)

## 7.2.- Gestionar el repositorio de los backups

Estamos copiando los datos en disco, y tenemos que decirle al sistema cual va a ser la carpeta principal bajo la cual guardará los datos. El valor por defecto es la carpeta **/var/spool/elkarbackup/backups**.

Si modificamos este parámetro una vez que hemos comenzado a realizar copias, el sistema no eliminará los datos antiguos, pero si intentamos acceder a ellos a través del botón Restore el sistema no los encontrará. Si por alguna razón necesitamos modificar este parámetro una vez que tenemos copias realizadas, deberíamos conectarnos a la consola del servidor y mover los datos de la ubicación antigua a la nueva.

Además, deberíamos asignar los permisos correspondientes en la nueva carpeta para el usuario y grupo elkarbackup

```
root@backups:~# chown -Rf elkarbackup.elkarbackup nuevopath
```

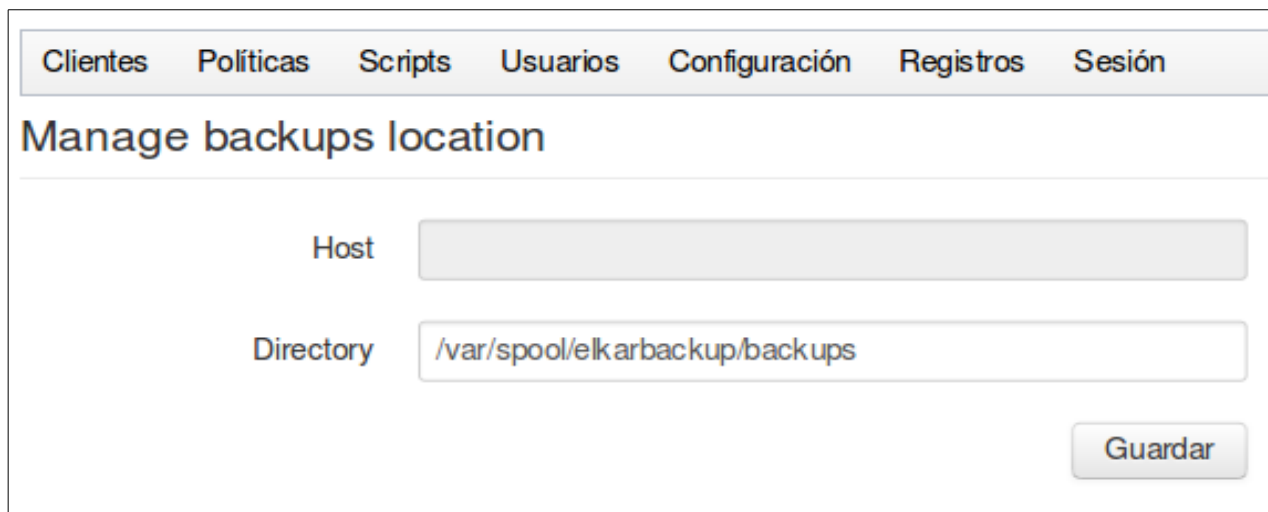
El disco que el servidor está utilizando para almacenar los datos puede estar conectado físicamente, o bien estar montado a través de la red, por ejemplo a través de los protocolos [ISCSI](http://es.wikipedia.org/wiki/ISCSI)<sup>9</sup> o [NFS](http://es.wikipedia.org/wiki/Network_File_System)<sup>10</sup>. Esta es una opción interesante cuando queremos montar ElkarBackup como servidor virtual.

La aplicación nos da la opción de definir cual va a ser el repositorio de las copias, y tenemos dos opciones:

- Disco local (dejando vacío el parámetro Host): Definimos un **directorio** local al servidor Debian en el que hemos instalado la aplicación ElkarBackup. Puede ser un disco físico o un disco montado previamente utilizando otras técnicas (por ejemplo a través de iscsi).
- Podemos configurar una carpeta de un servidor NFS (poniendo en el parámetro Host el Nombre/IPa del servidor). Para poder utilizar esta opción en el servidor Debian en el que hemos instalado ElkarBackup deberemos haber instalado el paquete **autofs** (nosotros lo hemos instalado al principio)

<sup>9</sup> <http://es.wikipedia.org/wiki/ISCSI>

<sup>10</sup> [http://es.wikipedia.org/wiki/Network\\_File\\_System](http://es.wikipedia.org/wiki/Network_File_System)



Es importante no olvidarse que si cambiamos la ubicación del repositorio el usuario **elkarbackup** de nuestro servidor Debian debe tener permisos de escritura en la nueva ubicación.

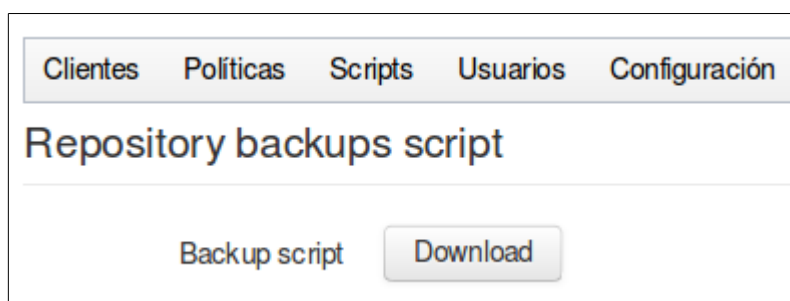
### 7.3.- Copia del repositorio

Con lo visto hasta hemos conseguido realizar copias de los datos de varios servidores, optimizando el espacio en disco y utilizando distintas políticas.

Pero ahora mismo tenemos todos los datos en un único disco, y si le pasara algo ya podemos ir diciendo adiós a nuestras copias.

Sería importante poder tener en algún otro sitio **una copia** del disco donde tenemos los datos, **en una ubicación física distinta** (para que no se viera involucrada en una situación de catástrofe), y que en la medida de lo posible que esta copia se sincronizara con la principal de forma automática.

En el menú podemos ver la opción **Configuración → Script de copia del repositorio**, y dentro tenemos la opción **Backup script descargar**.



Si hacemos clic en ese botón, descargaremos el siguiente script:

```
#!/bin/bash

MYSQL_DB=ElkarBackup
MYSQL_HOST=localhost
MYSQL_PASSWORD=root
MYSQL_USER=root
REPOSITORY=/var/spool/ElkarBackup/backups
SERVER=ElkarBackup
SERVER_USER=ElkarBackup
UPLOADS=/var/spool/ElkarBackup/uploads

ssh "$SERVER_USER@$SERVER" "cd '$REPOSITORY'; find . -maxdepth 2 -mindepth 2" |
sed s/^..// | while read jobId
do
    echo Backing up job $jobId
    mkdir -p $jobId 2>/dev/null
    rsync -aH --delete "$SERVER_USER@$SERVER:$REPOSITORY/$jobId/" $jobId
done
echo Backing up mysql DB
ssh "$SERVER_USER@$SERVER" "mysqldump -u$MYSQL_USER -p$MYSQL_PASSWORD
-h$MYSQL_HOST $MYSQL_DB" > ElkarBackup.sql
echo Backing up uploads
rsync -aH --delete "$SERVER_USER@$SERVER": "$UPLOADS/" uploads
```

Si ejecutamos este script en otra máquina (en adelante **Secundario**):

1. Se conectara al servidor donde tenemos instalado ElkarBackup y lanzará la sincronización de las copias de seguridad existentes.
2. A continuación realizará una copia de seguridad del servidor MySQL en el fichero ElkarBackup.sql
3. Por último también copiará los scripts que se hayan cargado en la carpeta **/var/spool/ElkarBackup/uploads**.

Por lo tanto sería suficiente programar y lanzar la ejecución de este script en el servidor **Secundario** para tener la copia sincronizada de nuestro repositorio en su propio disco.

Esta claro que para prever la situación de catástrofe, no tendría mucho sentido que ambos servidores estuvieran ubicados en el mismo sitio .....

## Automatización

Si queremos automatizar esta tarea, es decir, que el servidor **Secundario** sincronice de forma automática el repositorio del servidor **ElkarBackup**, es necesario volver a utilizar la estrategia de claves públicas/privadas.

En esta ocasión importaremos la clave pública del servidor **Secundario** en el servidor **ElkarBackup**, o dicho de otra forma, añadiremos la clave en el fichero **Authorized\_keys**. Para ello haremos clic en el botón **Añadir clave**.

Los campos a rellenar son dos:

- Comentario: Descripción que nos ayude a identificar la clave
- Clave: **Clave pública** del servidor Secundario

Para facilitar el proceso, crearemos en el servidor **Secundario** un usuario con el nombre **elkarbackup**

```
root@Secundario:~$ adduser elkarbackup
'elkarbackup' erabiltzailea gehitzen...
'elkarbackup' (1001) talde berria gehitzen...
'elkarbackup' (1001) erabiltzaile berria 'elkarbackup' taldearekin gehitzen...
'/home/elkarbackup' karpeta nagusia sortzen...
'/etc/skel'(e)tik fitxategiak kopiaitzen...
UNIX-pasahitz berria sartu:
UNIX-pasahitz berria sartu berriro:
passwd: pasahitza ongi eguneratu da
elkarbackup(r)en erabiltzaile informazioa aldatzen
Idatzi balio berria, edo sakatu 'Sartu' tekla lehenetsirako
  Izen osoa []:
  Gela zenbakia []:
  Laneko telefonoa []:
  Etxeko telefonoa []:
  Bestelakoa []:
Informazioa zuzena da? [B/e] B
```

A continuación abrimos una sesión con ese usuario

```
root@Secundario:~$ su - elkarbackup
```

y creamos su clave RSA

```
elkarbackup@Secundario:~$ ssh-keygen -t rsa
```



```

Generating public/private rsa key pair.
Enter file in which to save the key (/home/elkarbackup/.ssh/id_rsa):
Created directory '/home/elkarbackup/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/elkarbackup/.ssh/id_rsa.
Your public key has been saved in /home/elkarbackup/.ssh/id_rsa.pub.
The key fingerprint is:
1a:65:fd:52:08:05:89:8a:3a:89:68:23:72:c3:37:0e elkarbackup@portatil59
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      .o+.      |
|      . .o .    |
|     . . o o .   |
|    . . o  o    |
|o+   . S . .    |
|OoE o  o  .    |
|+o.=  ..       |
|      .        |
|               |
+-----+

```

Ahora podemos ver y copiar su clave pública

```

elkarbackup@Secundario:~$ cat .ssh/id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCUklE6TI16fU5pmTeU4APrSgG24eblwQdnHNtntUIIRSyAkAe
mPil2GDpufXgPKqT+FQV02z4JiaoTMMhQpsGqS/Shz/KE/MA7pm8k9v6qnFKVpY6HXZZyvgYhH+Yy6F
xxDk+QGQqQMnabzmanyxcBBdQ3ZdluYbwT5kdIgAJR8eTLN/M08hrKKeQGbEVXP3GCPWYsiDV2p6VgR
gkPzCSWUgMP63668ZAoNq8mlhW8RF+BEYDF9TPH7PJaEhc+Ea5LiiggD/E2lqQmFGYTqbjELKT4b97y
6nDj+2UIG0pvqZ/dN0ZMdsCMX577e8ppkafMqgwGT3D7Af4gD9KninL3 ElkarBackup@Secundario

```

y aprovechando la opción que tenemos en el interfaz web del servidor ElkarBackup, añadimos la clave en fichero **authorized\_keys** del servidor.



Comment / Key		
Servidor Secundario	ssh-rsa AAAAB3NzaC1yc2EAAA...	Borrar
		Guardar Add key

Pulsamos en el botón Guardar y clave importada.

Ahora, creamos una carpeta en el servidor Secundario para guardar los datos de la réplica, y copiamos dentro el script que hemos descargado desde el interfaz web

```
ElkarBackup@Secundario:~$ ls -la copiarepositorio/
guztira 12
drwxrwxr-x 2 elkarbackup elkarbackup 4096 eka 10 17:02 .
drwxr-xr-x 4 elkarbackup elkarbackup 4096 eka 10 16:53 ..
-rwxrwxr-x 1 elkarbackup elkarbackup 695 eka 10 15:24 copyrepository.sh
```

y comprobamos su ejecución

```
elkarbackup@portatil59:~$ cd copiarepositorio/
elkarbackup@portatil59:~/copiarepositorio$ ./copyrepository.sh
Backing up job 0002/0002
Backing up job 0001/0001
Backing up mysql DB
Backing up uploads
```

Como podemos observar hacemos la replicación sin tener que introducir ni usuario ni contraseña

```
elkarbackup@portatil59:~/copiarepositorio$ ls -la
guztira 128
drwxrwxr-x 5 elkarbackup elkarbackup 4096 eka 10 17:04 .
drwxr-xr-x 4 elkarbackup elkarbackup 4096 eka 10 16:53 ..
drwxrwxr-x 3 elkarbackup elkarbackup 4096 eka 10 17:04 0001
drwxrwxr-x 3 elkarbackup elkarbackup 4096 eka 10 17:04 0002
-rwxrwxr-x 1 elkarbackup elkarbackup 695 eka 10 15:24 copyrepository.sh
-rw-rw-r-- 1 elkarbackup elkarbackup 103785 eka 10 17:04 ElkarBackup.sql
drwxr-xr-x 2 elkarbackup elkarbackup 4096 eka 6 15:32 uploads
```

## 8.- Un repaso de conceptos: Rsnapshot

Tal y como hemos comentado anteriormente ElkarBackup es una herramienta basada en otras herramientas libres, sobre todo en rsync y rsnapshot, y como la mayor parte de su lógica responde a la lógica de rsnapshot, merece la pena detenerse para analizar su comportamiento.

Rsnapshot se basa en los siguientes conceptos:

- Origen de los datos a copiar: De donde tiene que copiar los datos ?
- Frecuencia: ¿Cada cuánto los tiene que copiar? En ciertas horas durante el día, una vez al día los días que nos interese, una vez a la semana y al mes el día que decidamos, ....
- Rotación: ¿Cuántas copias tiene que mantener en cada una de las frecuencias arriba mencionadas? Es decir, tal vez es suficiente con mantener las últimas 4 copias realizadas durante el día, pero las de final de mes las queremos guardar durante dos años.

A continuación intentaremos explicar su funcionamiento

### 8.1.- Frecuencia

Hay que definir cada cuanto tiene que hacer las copias, y estas son las opciones

- Durante el día (Hourly): Definimos las copias **que hará** en distintas horas del día. El sistema por debajo las nombra Hourly.0, Hourly.1, Hourly.2 .....
- Diarias (Daily): Definiremos las copias **que hará** cada día de la semana, especificando en qué días concretos y a qué hora las hará. El sistema por debajo las nombra Daily.0, Daily.1, Daily.2 .....
- Semanales (Weekly): Definiremos las copias **que hará** una vez por semana, especificando en qué día de la semana (sólo uno) y a qué hora las hará. El sistema por debajo las nombra Weekly.0, Weekly.1, Weekly.2 .....
- Mensuales (Monthly): Definiremos las copias **que hará** una vez al mes, especificando el día del mes y la hora. El sistema por debajo las nombra Monthly.0, Monthly.1, Monthly.2 .....

Cuando digo **que hará**, no estoy diciendo toda la verdad. Algunas veces hará las copias y otras rotará las copias previamente realizadas. Esto lo vamos a ver en breve al explicar la rotación.

## 8.2.- Retención

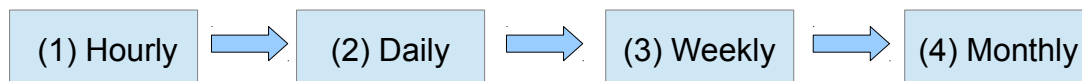
Con la retención vamos a especificar **cuantas carpetas** de cada frecuencia va a tener que guardar. Por ejemplo:

- Si ponemos una retención de 4 en las copias que se realizan durante el día (Hourly), eso significa que el sistema mantendrá las últimas 4 copias en las carpetas Hourly.0, Hourly.1, Hourly.2 y Hourly.3, donde Hourly.0 será la más nueva y Hourly.3 la mas antigua.
- Si ponemos una retención de 5 en las copias que realiza una vez al día (Daily), eso significa que el sistema mantendrá las últimas 5 copias en las carpetas Daily.0, Daily.1, Daily.2, Daily.3 y Daily.4, donde Daily.0 será la más nueva y Daily.4 la mas antigua.

Se aplica la misma lógica en las retenciones aplicadas a las copias semanales (Weekly) y mensuales (Monthly).

## 8.3.- Rotación

Esto no es algo complicado, pero hay que entenderlo bien. Al definir la frecuencia, también estamos estableciendo distintos niveles de copia, especificando de alguna forma que el Weekly **precede** al Monthly, que el Daily **precede** al Weekly, y que el Hourly **precede** al Daily. No hay ningún nivel que preceda al Hourly.



Podemos tener distintas políticas de copia, pero para el ejemplo imaginémonos que nuestra programación se compone de la siguiente manera:

- Las Hourly a las 11:00, 14:00 y 16:00, con una retención de 3
- Las Daily de lunes a viernes a las 21:00, con una retención de 10
- Las Weekly los sábados a las 21:00, con una retención de 5
- Las Monthly el 1 de cada mes a las 21:00, con una retención de 24

En este caso las copias se realizarían de la siguiente forma:

- Cuando llega la hora de ejecutar las Hourly primero borra la última (Hourly.2), y a continuación va cambiando el nombre del resto de carpetas. Hourly.1 → Hourly.2 y Hourly.0 → Hourly.1. Después realiza la nueva copia en la carpeta Hourly.0.

- Cuando llega la hora de ejecutar las Daily primero borra la última (Daily.9), y a continuación va cambiando el nombre del resto de carpetas. Daily.8 → Daily.9, Daily.7 → Daily.8, ..., Daily.0 → Daily.1 . La diferencia viene ahora, en este caso **no se realizará una nueva copia** para crear el Daily.0, se generará mediante una rotación de la última copia del nivel anterior: Hourly.2 → Daily.0
- Cuando llega la hora de ejecutar las Weekly primero borra la última (Weekly.4), y a continuación va cambiando el nombre del resto de carpetas. Weekly.3 → Weekly.4, Weekly.2 → Weekly.3, ..., Weekly.0 → Weekly.1 . En este caso **tampoco se realizará una nueva copia** para crear el Weekly.0, se generará mediante una rotación de la última copia del nivel anterior: Daily.9 → Weekly.0
- Cuando llega la hora de ejecutar las Monthly primero borra la última (Monthly.23), y a continuación va cambiando el nombre del resto de carpetas. Monthly.22 → Monthly.23, Monthly.21 → Monthly.22, ..., Monthly.0 → Monthly.1 . En este caso **tampoco se realizará una nueva copia** para crear el Monthly.0, se generará mediante una rotación de la última copia del nivel anterior: Weekly.4 → Monthly.0

Esto es lo que hay que entender: La rotación **solamente moverá la última copia del nivel anterior**, nunca moverá ninguna otra carpeta. Es decir:

- Si la retención del Hourly es 3, la rotación del Daily solo moverá el Hourly.2, nunca el resto.
- Si la retención del Hourly es 2, la rotación del Daily solo moverá el Hourly.2, nunca el Hourly.0

En esta lógica hay una condición que tenemos que tener clara; Nunca se tocan las carpetas que acaban en .0, y por esta razón **la retención no podrá ser 1 cuando en la política existan niveles superiores**, el Daily nunca tocará el Hourly.0, y el Weekly tampoco tocará el Daily.0 .

Es importante entender esto, ya que es la lógica que se utilizará por debajo cuando definamos las políticas desde el interfaz web.

## 9.- Anexos

### 9.1.- Tras descargara la imagen

Tenemos que tener en cuenta que todos los que descargamos esta imagen estamos utilizando la misma clave, lo cual puede suponer un problema de seguridad.

Desde el interfaz web en el apartado **Configuración → Gestionar parámetros** vemos el botón **Descargar** porque el sistema detecta que la clave ha sido previamente generada. Si por debajo borramos la clave, el interfaz cambiará el botón **Descargar** por el botón **Generar**, y tendremos la opción de generar una nueva clave

```
root@ElkarBackup:~# rm /var/lib/elkarbackup/.ssh/id_rsa.pub
```

### 9.2.- Script para comprimir el repositorio.

Imaginémonos que distintos usuarios tienen copias distintas del mismo fichero en la red, cada uno en su carpeta. Cuando se trata de documentos ofimáticos (generalmente no muy grandes) no suele suponer demasiado problema, pero con otro tipo de ficheros (vídeos, actualizaciones de software, etc) el sitio que pueden estar consumiendo en la red suele ser grande, siendo información duplicada.

Sabemos que una vez se ha realizado la copia de estos ficheros, estos no van a cambiar en el disco de la copia de seguridad, por lo que tenemos la posibilidad de mantener una sola copia y enlazar mediante hardlinks el resto de apariciones del mismo fichero.

En el ejemplo, copiamos y pegamos con otro nombre un fichero de los que tenemos en la carpeta **/media/Backups** del Cliente Debian. A pesar de ser el mismo fichero lo tenemos dos veces en el disco por lo que ocupa el doble.

```
root@DebianCliente:~# cd /media/Backups/Software/
root@DebianCliente:/media/Backups/Software# cp vlc-2.0.6-win32.exe vlc-2.0.6-
win32-kopia.exe
root@DebianCliente:/media/Backups/Software# ls -lah
total 209M
drwxr-xr-x 2 root root 4,0K jun 12 12:44 .
drwxr-xr-x 4 root root 4,0K jun 7 11:24 ..
-rw-r--r-- 1 root root 1,1M nov 18 2010 7z920.exe
-rw-r--r-- 1 root root 164M may 3 18:16 LibreOffice_4.0.3_Linux_x86_deb.tar.gz
-rw-r--r-- 1 root root 22M abr 15 19:11 vlc-2.0.6-win32.exe
-rw-r--r-- 1 root root 22M jun 12 12:44 vlc-2.0.6-win32-kopia.exe
```

Accedemos a la tarea encargada de hacer la copia de estos datos y pulsamos el botón **Ejecutar ahora**. Al finalizar la tarea podremos observar que en la carpeta del servidor ElkarBackup están los dos ficheros

```
#cd /var/spool/elkarbackup/backups/0001/0001/Hourly.0/media/Backups/Software/
# ls -lah
total 209M
drwxrwxr-x 2 elkarbackup elkarbackup 4,0K jun 12 12:44 .
drwxrwxr-x 4 elkarbackup elkarbackup 4,0K jun  7 11:24 ..
-rw-rw-r-- 3 elkarbackup elkarbackup 1,1M nov 18 2010 7z920.exe
-rw-rw-r-- 3 elkarbackup elkarbackup 164M may  3 18:16 LibreOffice_4.0.3_Linux_x86_deb.tar.gz
-rw-rw-r-- 3 elkarbackup elkarbackup 22M abr 15 19:11 vlc-2.0.6-win32.exe
-rw-rw-r-- 2 elkarbackup elkarbackup 22M jun 12 12:44 vlc-2.0.6-win32-kopia.exe
```

y como podemos observar no están enlazados mediante hardlinks, ya que no tienen el mismo **inode**

```
# ls -lahi
total 209M
40831 drwxrwxr-x 2 elkarbackup elkarbackup 4,0K jun 12 12:44 .
40828 drwxrwxr-x 4 elkarbackup elkarbackup 4,0K jun  7 11:24 ..
29332 -rw-rw-r-- 3 elkarbackup elkarbackup 1,1M nov 18 2010 7z920.exe
29333 -rw-rw-r-- 3 elkarbackup elkarbackup 164M may  3 18:16 LibreOffice_4.0.3_Linux_x86_deb.tar.gz
29334 -rw-rw-r-- 3 elkarbackup elkarbackup 22M abr 15 19:11 vlc-2.0.6-win32.exe
74101 -rw-rw-r-- 2 elkarbackup elkarbackup 22M jun 12 12:44 vlc-2.0.6-win32-kopia.exe
```

A continuación veremos como podemos utilizar un script postscript para resolver este problema.

Este script busca y enlaza mediante hardlinks aquellos ficheros que tienen el mismo **Hash**<sup>11</sup>

```
#!/bin/bash
# Compara por tamaño para descartar los que no se repiten

cd $ELKARBACKUP_PATH
lastHash=''
lastFile=''

find . -mount -type f -printf '%15s %p\0'|sort -nrz|uniq -zDw15|tr "\0" "\n"|
cut -b17- |tr "\n" "\0"|xargs$
do
    if [ "$x$lastHash" = "$x$currentHash" ]
```

11 <https://eu.wikipedia.org/wiki/Hashing> / [https://es.wikipedia.org/wiki/Funci%C3%B3n\\_hash](https://es.wikipedia.org/wiki/Funci%C3%B3n_hash)

```
then
    rm "$file"
    ln "$lastFile" "$file"
fi
lastHash=$currentHash
lastFile="$file"
done
```

Subimos el Script:

- Nombre: Compactar el repositorio
- Lo habilitamos para que pueda ser ejecutado como PostScript de tareas

Ahora editamos la tarea del Cliente Debian, seleccionamos este script en la sección **PostScript**, y lanzamos la ejecución de la tarea.

```
# ls -lahi
total 209M
40838 drwxrwxr-x 2 elkarbackup elkarbackup 4,0K jun 12 15:22 .
40835 drwxrwxr-x 4 elkarbackup elkarbackup 4,0K jun  7 11:24 ..
29332 -rw-rw-r-- 4 elkarbackup elkarbackup 1,1M nov 18  2010 7z920.exe
29333 -rw-rw-r-- 4 elkarbackup elkarbackup 164M may  3 18:16 LibreOffice_4.0.3_Linux_x86_deb.tar.gz
29334 -rw-rw-r-- 8 elkarbackup elkarbackup  22M abr 15 19:11 vlc-2.0.6-win32.exe
29334 -rw-rw-r-- 8 elkarbackup elkarbackup  22M abr 15 19:11 vlc-2.0.6-win32-kopia.exe
```

Si ahora nos fijamos en los números de inodo que aparecen en la primera columna, podemos observar que las copias aparecen con el mismo inodo, por lo que están apuntando a la misma ubicación en disco y no duplican el espacio utilizado.

### 9.3.- Ordenar tareas

Las tareas que pertenecen a la misma política se ejecutan todas una detrás de la otra. Cuando la ejecución de una tarea requiere mucho tiempo, puede retrasar de forma considerable la ejecución de las tareas que vienen por detrás. Se han detectado al menos dos casos en los que esto puede ocurrir:

- Cuando se hace la primera copia y el volumen de datos a transferir es grande.
- Cuando en una tarea hay muchos ficheros (en las pruebas 302000 ficheros).

El primer caso no debería preocuparnos porque será algo puntual. En cambio en el segundo caso, rsync debe leer un árbol de directorios grande tanto en el servidor como en el cliente, y esto le lleva tiempo. Ocurre lo mismo con las rotaciones de nivel inferior que utilizan operaciones "cp -al". Por último, la situación se repite al calcular el espacio ocupado en disco.



En estos casos tal vez preferiríamos que ciertas tareas se ejecutaran antes que otras, y para eso con el objetivo de ***asignar prioridades a las tareas*** se ha añadido la opción de ordenarlas a través de la opción del menú ***Clientes → Ordenar tareas***.

Para asignar las prioridades se nos mostrarán todas las tareas en una lista que podremos ordenar. Las copias se ejecutarán según el orden establecido en esta lista.

#### 9.4.- Transformando las imágenes a otros sistemas de virtualización

Si el sistema de virtualización admite imágenes en formato RAW no vas a tener demasiado trabajo, en otros casos necesitarás transformar la imagen al sistema que utilices.

Si utilizas Vmware aquí hay información que puedes utilizar:  
<http://wiki.laptop.org/go/VMware/Convert>

Nosotros hemos convertido la imagen con este comando:

```
:~# qemu-img convert ElkarBackupServerBase2GB1.0.9.img -O vmdk  
ElkarBackupServerBase2GB1.0.9.vmdk
```

Para VirtualBox puedes utilizar este otro comando:

```
:~# Deskargak$ VBoxManage convertfromraw ElkarBackupServerBase2GB1.0.9.img  
ElkarBackupServerBase2GB1.0.9.vdi
```

#### 9.5.- Sincronización con la nube

Hoy en día parece inevitable que al desarrollar una herramienta de copia de seguridad nos detengamos a pensar si hay que incorporar funcionalidades específicas para sincronizar nuestros datos con los servicios en la nube.

En esta aplicación no se ha desarrollado ningún apartado específico para esta función porque hemos visto que con los servicios que ofrecen este tipo de soluciones no hace falta desarrollar nada extra.

Si nos fijamos en el conocido Dropbox, vemos que tiene un tipo de instalación para poder ser [ejecutado como demonio](#) en un servidor Linux.

Si tenemos instalado Dropbox en nuestro servidor Elkarbackup, podemos desarrollar un PostScript para que mediante hardlinks enlace a la carpeta de Dropbox los ficheros que nos interesen de las tareas que queramos. Con esto ya tendríamos estos ficheros copiados en la nube de forma automática.

En este punto podríamos discutir sobre la conveniencia de tener nuestros datos en la nube, cuestiones como la privacidad de los datos, etc, pero bueno, eso lo dejamos para otra ocasión.

---

## 9.6.- Copias y Snapshots de los clientes Windows

Aunque no sea muy conocido, los sistemas Windows disponen desde hace ya tiempo de opciones para realizar snapshots. Gracias a esta opción podemos acceder y restaurar estados del sistema guardados automáticamente, algo que seguro has utilizado más de una vez para restaurar una situación anterior ante problemas de sistema.

[Por terminar]

## 9.7.- Disco NFS remoto

[Por hacer]

---

## 10.- Licencia

- Autor: Pedro Arreitunandia Ituarte
- Fecha: 2013/06/12
- Licencia: Creative Commons BY-SA: Reconocimiento – Compartir bajo la misma licencia:

<http://creativecommons.org/licenses/by-sa/2.5/es/>

