External Backup Script using Rsync

This script will create an archived backup in `.tar.gz` format and transfer the backup with rsync.

> This is an unofficial script that is provided for your convenience. The script is provided as-is and may not be updated or maintained by Ultra.cc. Customers are welcome to use and customize unofficial scripts for their unique needs and requirements. Unofficial support may be offered via Discord only and at the sole discretion of Ultra.cc staff. Use at your own risk.

First you will need to create a SSH key to be later imported to the destination host.

Creating Public and Private Keys

- On your slot, type in `ssh-keygen` and press **ENTER**. This should start generating public and private key pairs.
  - By default it generates a 2048-bit RSA key pair which is sufficient in most cases

```
$ ssh-keygen

Generating public/private rsa key pair.
```

- We also recommend to generating the following keys which are more secure than the default.
  - To generate an RSA 4096 key: `ssh-keygen -b 4096`
    - Much more secure than 2048 bit is slower when logging into your slot
  - To generate a ed25519 key: `ssh-keygen -t ed25519`
    - New algorithm, has a smaller key size and faster generation with security comparable to RSA ~3000
- Here, you can press **ENTER** to save the key pair into the `.ssh/` subdirectory in your home directory.

```
$ ssh-keygen -b 4096

Generating public/private rsa key pair.

Enter file in which to save the key (/your_home/.ssh/id_rsa):
```

- Here you optionally may enter a secure passphrase. You can press **ENTER** to skip putting in a passphrase
  - It is recommended to add in a passphrase
  - A passphrase adds an additional layer of security to prevent unauthorized users from logging in should they have your private key

```
$ ssh-keygen -b 4096

Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/your_home/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

- Then you should see the following output. You now have a public (`id_rsa`) and a private key (`id_rsa.pub`) stored in your Home folder (or on the path you set) that you can use to authenticate when logging into SSH.

```
$ ssh-keygen -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/your_home/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /your_home/.ssh/id_rsa.
Your public key has been saved in /your_home/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256: x23Tr+Ee5TlowA+U9HEquagnog3O09EYHQ346WY usbdocs@usbdocs
The key's randomart image is:
+---[RSA 4096]----+
|=.=.  .  o+..    |
|.B  o  .oo.      |
|o.o  oo  o       |
|.+ .  oo ...     |
| .. .  +S+ .     |
|.      =o==      |
|.o.    o.=o.     |
|o...  oE.+o      |
| .. .++..o.      |
+----[SHA256]-----+
```

Copying the new keys to the destination host

Type in `ssh-copy-id username@destinationip`

```
$ ssh-copy-id usbdocs@85.145.22.545
```

- The following output appears. This is normal. This means that your computer does not recognize your slot. This will happen the first time you connect to a new host. Type `yes` and press **ENTER** to continue.

```
$ ssh-copy-id destinationusername@85.145.22.545
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/usbdocs/.ssh/id_rsa.pub"
The authenticity of host '85.145.22.545 (85.145.22.545)' can't be established.
ECDSA key fingerprint is SHA256:9mQKWg1PVPZtxcDASEDdasawrqw.
Are you sure you want to continue connecting (yes/no)?
```

- Type in the SSH password for the destination

```
$ ssh-copy-id destinationusername@85.145.22.545
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/usbdocs/.ssh/id_rsa.pub"
The authenticity of host '85.145.22.545 (85.145.22.545)' can't be established.
ECDSA key fingerprint is SHA256:9mQKWg1PVPZtzZ6d5nDjcWUb/Flkuq5VHYRrvwTeRTE.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
kbguides@kbguides.lw902.usbx.me's password:
```

- Once you entered your password, OpenSSH will connect to the slot. It will then copy the contents of your `~/.ssh/id_rsa.pub` key into a file in your slot's home directory at `~/.ssh` called `authorized_keys`. Then you should see the following output. At this point, your `id_rsa.pub` key has been uploaded to the slot.

```


Number of key(s) added: 1


Now try logging into the machine, with: "ssh '85.145.22.545'" and check to make sure that only the key(s) you wanted were added.
```

- Once everything is done, you can login to your destination from your slot with just `ssh destinationusername@85.145.22.545` which will go straight to your shell securely. If you set a password for your keys, enter your password.

The Shell script

This Shell script will allow for an automatic backup of important information related to the configuration of your applications excluding Plex.. It can then be configured to store it or either google drive or a local folder on your Ultras Slot.

Initial Setup and Configuration

Your first step is to find the full path of your home directory

`pwd`

Something like this will be outputted be sure to make a note of it :

`/home1/usbdocs`

then create three new folders, one may already exist this is fine `mkdir ~/lock` `mkdir ~/scripts` `mkdir ~/autobackup`

Then you need to enter the new folder called scripts

`cd ~/scripts`

And create the script file

`nano backup.sh`

Paste the following lines into it :

```
#!/bin/bash
exec {lock_fd}>/home1/usbdocs/lock/BackupLock || exit 1
flock -n "$lock_fd" || { echo "ERROR: flock() failed." >&2; exit 1; }
if [ -z "$STY" ]; then exec screen -dm -S autobackup /bin/bash "$0"; fi

DATE=$(date +%Y-%m-%d-%H%M%S)
tar --exclude="$HOME"/.config/plex -czvf "$HOME"/autobackup/MyUSBbackup-$DATE.tar.gz
"$HOME"/.apps "$HOME"/.config



rsync -aHAXxv --numeric-ids --info=progress2 -e "ssh -p portnumberhere"
/home1/usbdocs/autobackup username@destinationip:/home/username/destination/folder



flock -u "$lock_fd"
```

Save it by pressing `Ctrl+X then Y then Enter`.

You will need to change the paths `home1/usbdocs` to match your own home and username. You will also be required to change `destinationip` to match your destination

Testing

So to test first we navigate to ~/scripts folder we made earlier

`cd ~/scripts`

Then we need to allow the backup.sh permissions to run

```
chmod +x backup.sh
```

And finally, run it

```
./backup.sh
```

If all went well * tar -ztvf ~/autobackup/*.tar.gz* should show a bunch of files starting with /home/username/.config and /home/username/.apps inside.

If the script is running and you were to rerun it, you may see an error message “Flock Failed” this is a file lock to stop multiple downloads running and is normal. If you are sure it isn’t running you can delete the lock file from ~/lock. You can also check the progress of the backup script, which is running in a screen with the command `screen -rd autobackup`

If all is well after the test, we can automate the check via crontab

Open crontab with

```
crontab -e
```

You may have a choice of editors. We recommend Nano

Inside the crontab add a single line under everything else in the file that looks like this

```
0 0 */3 * * /home/usbdocs/backup.sh
```

Save it by pressing Ctrl+X then Y then Enter.

The script will now run every 3 days, checking for files that have changed and syncing them to the destination folder

---

Revision #8
Created 14 July 2021 17:49:35 by Joe
Updated 1 September 2023 18:00:22 by varg