installing RUBI for making backups

RUBI is Beat Rubischons “Rubi’s Ultimat Backup Implementation” and it’s indeed quite a clever thing ;)

here’s what it does, or what i think it does ;) every time you run rubi, it copies files from a source to a destination. therefore it creates a new folder in the destination drive containing the current date and time in the directory name. it then copys all files from the source into that new directory. but if it did just regular copies, it would cause quite some disk usage, so rubi came up with a clever solution. all the unchanged files are hardlinked to the previously downloaded copies. this way, we only need the disk space once, but we still have all those directories where we have a whole snapshot of every day. No more messing around with incremental stuff :)

the only drawback i found so far.. there’s no documentation..

so here’s how i got it up and running.. but be aware, im not really a linux expert ;)

first, create a new home for rubi. i suggest to install it into /opt so we don’t need to create a directory at all :) then download and unpack rubi. it can be downloaded from the rubi section of beat’s webpage by the time i wrote this article, the most recent version was 3.1.0. always check for updates before copy/pasting my lines!

```
cd /opt
wget http://www.0x1b.ch/misc/hacks/rubi/rubi-3.2.0.tar.gz
tar xzvf rubi-3.2.0.tar.gz
rm rubi-*
```

now we should also install some dependencies and find out where they are installed. the defaults will probably mostly work.

rubi needs the folowing shell tools: cpio, find, mkfifo, rsync and ssh or rsh if you want to backup through a network rather than only from one harddisk to another on the same server. these tools are mostly system essentials which you probably won’t need to install.. except for rsync.. on debian for example run

```
apt-get update
apt-get install rsync
```

using whereis you can find out where these tools are located so we can tell rubi

```
whereis cpio
```

for example will tell you where to find cpio

note down all tose paths, we need them for the configuration of rubi

now since there is no documentation to rubi, beat has simply included many examples of how he did things, so we can just pick the one that fits our needs the most and modify it… all examples are in `/opt/rubi/bin/rubi-xy` that’s the place for our final configuration of ruby to be too.

```
cd /opt/rubi/bin
cp rubi-test rubi
vi rubi
```

now let's configure it the way we want it first the global settings.. just make sure all the path's are correct and uncomment the right remote shell tool for your system.

you don't need to change anything to the source shell functions section ...

and now the most important part.. our actual backup setups... here's an example of what it looks like:

```
# ------
# Backup of pssrv
# ------
SRCHOST=192.168.168.43                          # source host
SRCPART="/"                                     # source partitions
BASEDST=/mnt/suter/psuter_backup/psbackup       # base Backup dir
DSTDIR=$BASEDST/pssrv_`date +%Y.%m.%d-%H%M`     # destination
OLDDST="`cat $BASEDST/lastdst 2>/dev/null`"     # last destination
SRCCPIO=/bin/cpio                               # remote GNU cpio
SRCRSYNC=/usr/bin/rsync                         # remote rsync
KEEPD=10                                        # keep 10 days
KEEPS=                                          # keep string
UTIME=1                                         # set utimes

doit | tee $DSTDIR.log
```

if you want to do several different backup sets right after each other, simply put several of these blocks at the end of your rubi file. Please make sure to use different BASEDST for each of these blocks (the reason why can be seen a little further down in the text at the BASEDST description).

SRCHOST is the source host name. if you do local backups, thats localhost, otherwhise it's an ip or hostname of a remot computer and you will need rsh or ssh and rsync if you have larger files than 2gig

SRCPART is a list of directories you want to backup. rubi will descend in all subdirectories of these derecotires listed here... the single directories are separated with a spacer and it has to be inside quotes. so it could look like this:

```
SRCPART="/dir1 /dir2 /dir3"
```

this will create a backup of `/dir1`, `/dir2` and `/dir3` into the BASEDST directory

BASEDST is the directory that will hold all those timestamped subdirecotires of each backup. You should provide a different BASEDST for each backup-set you have, because RUBI writes a file named `lastdst` into BASEDST where it stores the name of the last complete path (including the timestamp) where the backup was made to. it will then compare the new files with those and decide if it should link or copy the files. if you have the same BASEDST for two Backup-Sets, this file will be overwritten each time a block completes and so RUBI will always compare the wrong directories...

DSTDIR is the definition of how a timestamped directory name is being put together

OLDDST as far as i understood it, that’s the place where a timestamped directory goes before it is being erased after it expired

then you can tell where cpi and rsync are to find on the source server (needed when it’s not in the same place on a remote server)

KEEPD sets how many days a backup should be kept befor it is being erased again

the other two opitons i don’t understand yet.

don’t forget tho put the doit line after the config block.. that's what starts the backup

if you want to do anything else, check out the different examples from beat.

now to run your backup simply run

```
/opt/rubi/bin/rubi
```

if you want to automatically backup your data, create a cron job for it either by using

```
crontab -e
```

or by editing the main crontab directly

```
vi /etc/crontab
```

NOTE: you need to get ssh autologin to work. so make sure your backup machine's public ssh key is in the authorized_keys file on each target machine, otherwhise your backup won't run automatically as the backup server won't be able to login.

addon: backup from or to a qnap NOTE *the description that followes below was written before Qnap supported Docker containers. Nowadays I recommend to simply setup a docker container and run your linux stuff in there rather than hacking it into the qnap OS as the docker method will remain functional evena after firmware upgrades. I leave this here just for reference.*

a little addon to install rubi on a qnap or to backup from a qnap (only on intel versions, as there seems to be no ipkg repo for the ARM based ones)

install ipkg on the web gui

```
ipgk install cpio
ipkg install nano
```

(if you prefer nano over vi)

```
ipkg install coreutils
```

(onyl necessary if you plan to backup to this qnap, it’s not netcessary on the source i think)

```
ipkg install perl
```

download and copy rubi to your harddisk (/share/HDA_DATA/.rubi) and edit the two .pl files in the lib/

subdirectory. change the path of perl in the first line from /usr/bin/perl to /opt/bin/perl

```
crontab -e
```

to enter the crontab entry

to keep the whole installation running EVEN after a reboot ;) be careful, as far as i understand the box (which i don't own and don't have access to the web interface to) it should automatically be mounted. however, on one of two boxes this did not work out, so here is a shell approach to hack it.. but maybe some playing around in the ui might help as well.. just make sure that the optware stuff is autostarted and it should theoretically work :) otherwhise: on the qnap do the follwing...

```
mount /dev/mtdblock5 -t ext2 /tmp/config cd /tmp/config vi autorun.sh
```

(it may as well be mtdblock4 or 6, depending on your qnap box, just try which one will mount)

add the follwing contents to the file:

```
rm -rf /opt ln -sf /share/HDA_DATA/.qpkg/Optware /opt echo "export
PATH=$PATH:/opt/bin:/opt/sbin" >> /etc/profile
```

make sure your harddisk is HDA_DATA, it may also be something else..

make it executable and unmount the partition.

```
chmod +x autorun.sh cd / umount /tmp/config
```

now reboot to see if it works :)