attached our deck of scripts, with some verbiage of how to use them.

feel free to modify them as you see fit.

we use them from Linux box (to some extent, they can also be used from windows cygwin environment)

there are a number of packages needed to be installed for them to work,  
nothing too hard to find or install.

you're welcome to have a look at the script, use them and suggest modifications.

note - those are lab level scripts, not product level, **use them with appropriate care**.

if you wish to review them with me, or have an overview or whatever,   
we can setup a phone call to discuss them.

last note - I the scripts written such that it's assumed they all in the same folder, and are in the system execution path.

I recommend either put them in a folder that is in path (or add the folder to)

or at least add it on a temp basis when preparing to work with it.

**I. single miner scripts**

**1. updateMinerImage**- update image on miner.

Usage: updateMinerImage <ImgFile> <MINER\_IP> [-check | -force]

replace image on miner and reboots miner.

validates img file is legit, and replace it on the miner.

-check prompts for confirmation before apply.

-force will replace image even if identical version id.

default - suitable for mass script, doesn't prompt, and skips in case of identical version.

This script takes a uImage file (not that tar wrapped format available in the firmware server)

2. **updateMinerSoftware.sh**  
usage: updateMinerSoftware.sh <IP> <firmware tar package>  
update image on the miner, similar to updateMinerImage (and updateMinerCaller)  
only that it deploys firmware from a tar file packages, as reside on our firmware server.

**For the ultimate method of updating miner images – see**remote\_local\_update.sh

3. **miner**- wrapper for ssh into miner  
usage: miner <last ip nibbles>

currently wraps calls using prefix 10.0.0 - change to what appropriate.

convenient when doing ssh to machines on minute to minute basis.

The prefix resides in the **constants** file – edit this file to suit your network layout.  
  
4. **pingver** - pings a miner and upon IP accessibility gets its version id.

usage: pingver <miner\_IP> [-delay delay-sec]

the big advantage here that it has very little dependency in the miner

running state etc, as it relies only on ping and ssh.

Very useful for basic mapping of data center (when called from mass scripts).

Use the –delay option (look at pingver\_slow for an example)   
in cases of slow connection between monitor and miners' LAN.

5. **cpfile**- copy a file to miner, and restart miner application  
usage: cpfile <miner\_IP> <file\_to\_copy> <destination\_in\_miner\_filesystem>

very convenient for configuration change. e.g. cgminer.conf change would be done

as following:.

carve cgminer agreeable configuration on miner.

download it using scp , store it e.g. at ~/configs/cgminer.conf-goodandsold

then perform

cpfile <Miner\_IP> ~/configs/cgminer.conf-goodandsold /etc/cgminer.conf.template

as we'll see in a few lines, this works also for mass copy!

Notes:  
I. This script restart mining post copying. For a "non-restarting" version use cpfile\_norestart script.

II. for better script deploying cgminer.conf (deals with expansion of %h etc, and supports post 2.5.53 template files) use replace-wallet-file.  
III. For altering cgminer.conf – have a look at allow-cgminer-remote-mon  
  
6. **mon** - calls wget .. monitor.php

usage: mon <miner\_IP>

wrapper for calling wget http://admin:admin@<MINER\_IP>/monitor.php

provide miner stats. (JSON format).  
useful in scripts, and also in console ,

when piped into jq .  
  
7. **minerstat** - output miner stats

usage: ./minerstat <miner\_IP> [1]

output csv format of miner stats.

1 in extra parameter will cause header line to be produced.

(very usefull for mass call, which creates a csv spread sheet in one call)

Note – minerstat relies on miner monitor.php page which relied on cgminer API. Thus, will work only when cgminer works. i.e. it will not work when mining is stopped.

8. **ssher** - generic ssh call into miner

usage: ssher <miner\_IP> cmd ...

all parameters passed the IP are passed via ssh. however, sometimes it's better to send them in parenthesis so that expansion is done on target and not on caller.   
or to allow piping of commands

e.g.

ssher <miner> "ps | grep miner"

this allows also passing small scripts using ;  
e.g.  - when preparing miner for decomission  
we would like to clear it's configuration and light the leds then we would send  
ssher <miner> "/usr/local/bin/spond-manager stop ; rm -f /mnt/config/etc/\* ; rm -f /mnt/config/rrd/\* ; rm -f /mnt/config/log/\* ; /usr/local/bin/leds & "

**note**- when calling an ssh command, there is no environment with it, so path env variable is not as when logging into machine. make sure appropriate way of calling command is used.

9. **pushndo.sh**– deploy a script on target miner, and call it

usage: pushndo.sh <miner\_IP> <trg\_scriptfile> [params …]

copies a script file onto miner (into folder /usr/local/bin ) and call it on the miner.  
note -   
the script will remain on the targets /usr/local/bin folder (until manually erased or next reboot)

**II. group calling scripts**1. **apply\_on\_ips.sh**  - calls script on list of miners  
usage: apply\_on\_ips.sh <script> <IP\_LIST\_FILE> param1 ...

this script reads a list of IP from a file (see comment on IP list) and  calls each IP (miner)

the script with the parameters.

<script> should be given in an executable path. i.e. either absolute, or correct relative or that it's in search path already.

param1 - at least one parameter must be passed. it may happen that the script requires no param. then pass a dummy parameter.

This is VERY useful for mass operation.

examples:

apply\_on\_ips.sh minerstat <MINER\_IP\_LIST> foo | tee ~/miners.csv

will create a csv file with miners stats.

apply\_on\_ips.sh pingver <FULL\_DHCP\_RANGE\_IP\_LIST> | grep -v NO-PING | grep -v SSH-ERR | tee ~/MINERS\_IP\_LIST

(a bit slow but) an easy way to generate an IP list

apply\_on\_ips.sh ssher <MINERS\_IP\_LIST> "/sbin/reboot &"

**2. sharsheret –** generic ssh call for multiple miners

Usage: sharsheret miner1-IP [miner2-IP ... minerN-IP] "ssh\_command [ parms ... ]"this is a convenient extension of ssher to a set of multiple miners separated by space.  
this allows one call per many miners, and also more flexible, and doesn't require an IP table file to iterate. i.e. similar to "apply\_on\_ips.sh ssher" but more flexible (can work on one miner, a few miners or even a IP table file.  
for example  
sharsheret $(cat /home/miners/scan-0125.csv | grep SP30 | cut -d, -f1) "/usr/local/bin stop &"

This command will send stop mining command for all the SP30 miners, that are found in the scan file.

Note – ssher assumes one miner and treats the rest of the line as command and parameters to be passed with the ssh command.  
sharsheret assumes at least one miners but potentially many, and only the last part of the line is the ssh command. Thus, passing of command with parameters must be put in quotation marks.

3. **inpar** - calling apply\_on\_ips.sh but in parallel.

a bit coarse script, but very useful when doing tasks that are time consuming, such as updating images.

while updating image using

apply\_on\_ips.sh updateMinerCaller <MINER\_IP\_LIST> <uImageFile>

will work just fine, it will take a long while for large set of miners

another example would be a scanning of a wide range of IP addresses. the scanning performs ping and ssh, which for large ranges would take a long time (as a dead IP would take a second to timeout). using inpar, the discovery would take a short fixed time because the list is split to lists of 20 addresses.

inpar also collect log to split files in template of MINER\_IP\_LIST\_??.log

those can then be cat together into one central file, thereby consolidating the striped ouput to one.  
Usage: inpar [-show] [-pile <size>] <scriptfile> <ipfile> params ...

4. **updateMinerCaller**

this is a special wrapper script we use to call updateMinerImage from the list based caller scripts (apply , inpar)  
the reason is that in the updateMinerImage the order of parameters is different, and we didn't want to break it, so the updateMinerCaller script simply change the params order and call the other one.

5. **remote\_local\_update.sh**

Usage: remote\_local\_update.sh <IP-File> <fw\_version>  
e.g. remote\_local\_update.sh /home/miners/DC-All 2.6.11

The script will look for the fw of version 2.6.11 in the specified (in constants) image storage folder. If it's not there it will download the tar file from the web,  
then it will push the tar file to one of the miners, and call all miners in the list to upgrade from it. This is **VERY** convenient way of updating mass of miners.  
Also – this method also is very efficient when network between miners and monitor is not very fast. The fw package is uploaded to the miners from the monitor only once, and the rest is done on the miners' local network!  
Note – use this script on a group of co-located miners. i.e. if you have multiple DCs monitored, apply it for each DC separately.

6. **createdetected**create a list of detected miners (within a range of IPs)  
usage: createdetected <wide\_range\_ip\_list> [nofile | output-filename]

the range list is typically the entire relevant range of addresses which miner may get from dhcp etc.i do it with the entire DHCP range of the datacenter.

nofile would send output only to screen

if no output file name given, a time based file is created - the name printed in the output.

7. **milist** create a list of miners based on subnet  
usage: milist <base\_subnet> <number\_of\_segments>   
e.g. milist 192.168.0.0 11 will scan from 192.168.0.0 thru 192.168.10.255  
the script will create a list, put both on screen and in a file, which includes the miners detected within the IP range specified in the command line.

8. **perform\_scan.sh** create csv file with miners stats

usage: perform\_scan.sh <MINER\_IP\_LIST> [nofile | outputfile] [-slow]

rely on miners monitor.php interface  
creates a csv table (with header row) with miners details and operation stats.  
very convenient for immediate insert into excel,  
and for piped processing in command line (grep,cut,awk etc).

nofile would send output only to screen

if no output file name given, a time based file is created - the name printed in the output.

The –slow switch helps in case the monitoring laptop is connected with substantial latency to the miners.

6. **monitor\_group.sh**

perform perform\_scan on a list of IPs multiple times ,with set number of repeats and delay.

nice for an overnight or such close monitoring of a group.

**III. miner targeted scripts**These are scripts we commonly use with **pushndo.sh** (see above)  
these are scripts that are run on the miners themselves (not on the monitor!)  
we push them to the miner via pushndo.sh script.   
Usually, these scripts are such that are either too complex to be conveniently written in a one line script passed as ssh command, or such that are inherently impossible (e.g. changing network config), or that are so commonly used that it makes sense to encapsulate them as flied scripts.

1. **trg\_miner\_power.sh**– output hash voltage and power date

2. **trg\_cleanup.sh –** cleans all non-volatile file system except for mining configuration (can be easily altered to "factory reset").

3. **trg\_set\_mg\_custom\_mode\_field.sh** modifies a numeric field of the mg\_custom\_mode file. For example trg\_set\_mg\_custom\_mode\_field.sh VMAX 690 would set the max voltage field to 690 (without changing other fields).

4. **trg\_miner\_deep\_conf.sh** - similar to trg\_miner\_power.sh  but provides more configuration information.

5. trg\_redo.sh similar to trg\_cleanup,sh only deeper.

6. trg\_join\_monitor demonstrate how to allow miner's cgminer API being monitored by an external server (important for integration with monitoring apps that work via cgminer API).

**Notes:**

1. **Constants file:**  
holds global common constants and a few common alias and util functions. Edit it to reflect specific folders (for images, for scans) or passwords if you changed them, etc.

2. **IP list**  
we expect a list of either just ips in the file, one in a row,

or a file either space separated or comma separated, where IP is the first field.  
the scripts that read

typically we extract the image from it, so it can be used in the script.

**packages dependencies:**

1. dos2unix

(needed to convert IP address list, in (typical) case they are imported from windows.

leaving  list in windows format, will make reading it more complicated.

2. sshpass - allows ssh connection without prompt

3. uboot-tools. we use mkimage from uboot-tools to validate image file.

4. jq (JSON command line parser) available at <http://stedolan.github.io/jq/download/>

**File List**apply\_on\_ips.sh

checkDupVPD.sh

constants

cpfile

cpfile\_norestart

createdetected

createdetected\_slow

crossfile

dc-many-scans.sh

findminertags.sh

findwallet.sh

inpar

milist

miner

minerstat

mon

monitor\_group.sh

perform\_scan.sh

pingver

pingver\_slow

pushndo.sh

remote\_local\_update.sh

replace-cgminerIP

replace-wallet-file

rmfile

sharsheret

ssher

trg\_cleanup.sh

trg\_join\_monitor

trg\_miner\_deep\_conf.sh

trg\_miner\_power.sh

trg\_redo.sh

trg\_set\_mg\_custom\_mode\_field.sh

update\_caller

updateMinerCaller

updateMinerCaller2

updateMinerImage

updateMinerImageAndClean

updateMinerSoftware.sh

validateMinerImage

(updated for package miner\_scripts-20150401.tar)