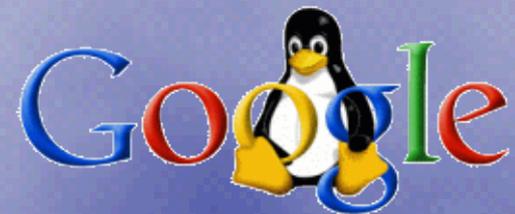


Linux.conf.au 2004

Maintaining lots of apt-get  
capable linux machines  
with apt and getupdates

Marc MERLIN

marc\_soft@merlins.org



# What you don't want

- Each sysadmin installed a server from some install CD. (If you're lucky, it might be the same CD, but if not, it might be different releases, or even worse, different distributions)
- Each server has local, unknown, customizations from each sysadmin
- You have 300 workstations with an old version of some distro, and you have to upgrade sshd, or convert them to autofs with ldap
- Install server? What install server?

# Challenges

- You mean I have to maintain workstations where users have root and can install and remove random software?
- Why maintain servers and workstations separately?
- But how do I keep all this in sync when all the machines don't even have the same list of software installed?
- How do I even update software? How about conf files?
- Do I need to change the install each time I make changes to the installed base?
- How about machines that were loaded before I changed the install but after I synced the change to the installed base?

# Non working solutions I've seen

- All workstations rsynced from central image
  - very heavy on the rsync server(s)
  - nightmare to maintain rsync includes/excludes
  - unsafe to upgrade libc
  - hard to maintain local software
- chroot live image with ssh/scp pushes
  - much better (install & clients in sync)
  - support for local software
  - disaster if some clients can be down since they miss updates
- nice install server with no updates
  - No comments...
- nice install server with push updates
  - what about clients that are down during updates?
  - install out of sync with pushed updates

# Requirement for a good solution

- Should work for several class of clients (servers, light servers, workstations...)
- clients should always end up being up to date (replay updates in order if you were down)
- if an update fails, it will retry later
- an update failure should not put a client into an unknown state it may not be able to recover from
- The install should be updated at the same time than the installed base, and there should be no lapse of time for new installs
- Some downgrade/rollback capability

# Getupdates bonuses

- Initial install can be on CD/DVD, nfs/ftp/http, or made with an out of date disk copy
- clients are up to date as soon as they reboot after the install
- lead and gold (i.e. unstable vs stable)
- you know about all your clients:
  - cpu/memory/disk/hardware information
  - linux kernel version
  - list of packages installed
  - all users logged in the machine present and past / machine “owner”
  - convert between hostname, IP, and MAC address (useful when the client is down)

# Implementation: Basics

- apt-get takes care of installing packages and their dependencies (computed automatically) via nfs/http/ftp
- changesets are comprised of a shell script and a compressed tar archive containing optional files used by the update (just two files to retrieve, which is easier for http/ftp)
- getupdates is the core of the install and update mechanism and pulls changesets via nfs/http/ftp.

getupdates runs at install time and from cron, checks the update server for a list of changesets for the client's \$TARGET, and retrieves the update if it's new, or if it's been modified since the last time it was run.

# Implementation: Variables

## ➤ /etc/sysconfig/getupdates:

`TARGET=wkslead`

`NFSHOST=software.nfs`

`NFSSWPATH=/software -> $NFSHOST/$NFSSWPATH (boot media) or $NFSSWPATH (autofs)`

`PROTOCOL=nfs`

`WORKDIR=/var/lib/grhat`

`STATBASE=/auto/clientinfo -> runtime (autofs)`

`NFSSTATMNT=clientinfo.nfs:/clientinfo -> load time from boot media`

- `$TARGET` determines lead vs gold and wks vs srv
- `clientinfo` is NFS only and for stats and logs
- `$PROTOCOL` is current http or nfs, and can be ftp easily

# Implementation: Layout

- File layout in the install tree:

```
screen
Eterm Font Background Terminal
stan:/software/grhat/targets# ls -l
total 20
lrwxrwxrwx 1 root root 54 Nov 13 16:12 Makefile.changeset -> /home/build//ops/corp/grhat/scripts/Makefile.changeset
-r----- 1 merlin eng 510 May 19 2003 Makefile.tagwork
drwxr-xr-x 7 merlin ops 4096 Nov 18 11:28 srvgold
drwxr-xr-x 6 merlin ops 4096 Sep 30 13:18 srvicead
lrwxrwxrwx 1 merlin ops 7 Apr 23 2003 srvttest -> srvicead
drwxr-xr-x 19 root root 4096 Sep 29 21:33 wksgold
drwxr-xr-x 19 root root 4096 Oct 10 09:35 wkslead
lrwxrwxrwx 1 merlin ops 7 Apr 23 2003 wkstest -> wkslead
stan:/software/grhat/targets# cd wkslead; ls -l
total 72
drwxr-xr-x 3 root root 4096 Sep 29 09:34 00050_PwdSudo_Mks
drwxr-xr-x 3 root root 4096 Nov 12 23:26 00100_RedHatPatch1
drwxr-xr-x 3 root root 4096 Sep 18 16:22 00110_ScriptsInstall
drwxr-xr-x 3 root root 4096 Nov 26 12:50 00120_GoogleConf1
drwxr-xr-x 3 root root 4096 Nov 21 13:55 00160_LinksInstall
drwxr-xr-x 3 root root 4096 Oct 2 13:12 00200_AptBatch
drwxr-xr-x 3 root root 4096 Oct 17 08:16 00220_AptBatch_Mks
drwxr-xr-x 3 root root 4096 Oct 13 15:32 00240_AptLangModules
drwxr-xr-x 3 root root 4096 Nov 20 16:08 00320_Kernel_Mks
drwxr-xr-x 3 root root 4096 Nov 15 20:06 00500_AptUpgradesnOtherInstalls
drwxr-xr-x 3 root root 4096 Nov 14 09:47 00510_AptUpgradesnOtherInstalls_Mks
drwxr-xr-x 3 root root 4096 Jun 27 09:57 09999_AptGetUpgrade
drwxr-xr-x 3 root root 4096 Aug 8 19:03 10110_FixMksPackages1
drwxr-xr-x 3 root root 4096 Aug 14 00:23 90000_FixPerms
drwxr-xr-x 3 root root 4096 Nov 11 15:45 95000_Obsolete
drwxr-xr-x 3 root root 4096 Oct 10 09:35 99000_WallReboot
drwxr-xr-x 2 root root 4096 Aug 28 21:49 links
-rw-r----- 1 root root 324 Nov 21 11:08 list
lrwxrwxrwx 1 root root 19 May 19 2003 Makefile -> ../Makefile.tagwork
```

```
screen
Eterm Font Background Terminal
stan:/software/grhat/targets/wkslead# head -3 list
00050_PwdSudo_Mks
00100_RedHatPatch1
00110_ScriptsInstall
stan:/software/grhat/targets/wkslead# cd 00110_ScriptsInstall; ls -l
total 40
drwxr-xr-x 4 root root 4096 May 12 2003 files
-rwxr-xr-x 1 root root 107 Jun 2 20:39 fixperms
lrwxrwxrwx 1 root root 24 Jun 12 06:40 Makefile -> ../../Makefile.changeset
-rwxr-xr-x 1 root root 1445 Nov 25 18:15 runme
-rw-r----- 1 root root 11 Nov 30 19:28 updatedate
-rw-r----- 1 root root 26806 Nov 25 18:15 workfiles.tar.bz2
stan:/software/grhat/targets/wkslead/00110_ScriptsInstall# cat updatedate
1070249338
stan:/software/grhat/targets/wkslead/00110_ScriptsInstall# make newdate
Done
stan:/software/grhat/targets/wkslead/00110_ScriptsInstall# cat updatedate
1070250806
stan:/software/grhat/targets/wkslead/00110_ScriptsInstall# find files -type f | head -3
files/var/lib/grhat/funcs
files/usr/local/scripts/corphost
files/usr/local/scripts/copysafe
stan:/software/grhat/targets/wkslead/00110_ScriptsInstall# cd ../links; ls -l
total 0
lrwxrwxrwx 1 root root 49 Jun 13 19:33 funcs -> ../00110_ScriptsInstall/files/var/lib/grhat/funcs
lrwxrwxrwx 1 root root 58 Jun 13 19:33 getupdates -> ../00110_ScriptsInstall/files/usr/local/scripts/getupdates
lrwxrwxrwx 1 root root 48 Jun 13 19:33 shlock -> ../00100_RedHatPatch1/files/usr/local/bin/shlock
stan:/software/grhat/targets/wkslead/links#
```

# Implementation: Postinstall

- You need to bootstrap getupdates via HTTP/FTP/NFS
- We read config options from /proc/cmdline: options are passed to install kernel (via syslinux / other boot loader)
- Some getupdates logic in postinstall to get load options, use them to retrieve getupdates and its tools, install them on disk, and save boot options in /etc/sysconfig/getupdates
- Distribution specific, but needs to be run in a chroot after the distro's install has completed (%post in a Red Hat kickstart file)
- Install getupdates ASAP so that it can continue the install if machine crashes or gets unplugged during postinstall

# Implementation: Postinstall Code

```
screen
Eterm Font Background Terminal
log "=== Start Postinstall ==="
mkdir /usr/local/scripts
mkdir /var/lib/grhat

> /etc/sysconfig/getupdates
# Lets you specify the domain if you need to with !! instead of the whole
# thing (saves on the characters in the char buffer for kernel arguments
DOMAIN=corp.google.com

# Those variables are used both here and inside getupdates once the machine
# is up and running, so we read them here and save their values in
# /etc/sysconfig/getupdates if they are non default
# TARGET is something like wksgold
# NFS loads are made from $NFSSHST:$NFSSUPDPATH:$TARGET/
# HTTP loads are made from http://$HTTUPDPATH:$TARGET/
# PROTOCOL defaults to http but you can set to nfs
for VAR in TARGET NFSSHST NFSSUPDPATH HTTUPDPATH TARGET PROTOCOL
do
  var=`echo $VAR | sed 's/\(.*\)/\1/'`
  VALUE=`cat $CMDLINE $CMDLINE2 | grep "$var=" | tail -1 | sed -e "s/.*$var="/`
  if [ -z "$VALUE" ]; then
    eval `echo "$var=$VALUE"`
    log "Got $VAR=$VALUE from kernel command line"
    echo "$VAR=$VALUE" >> /etc/sysconfig/getupdates
  else
    log "Didn't get $VAR from kernel command line"
  fi
done

if [ -z "$TARGET" ]; then
# Note that because of our krb setup, if the install dies in the wrong
# place (and that window is big), it is likely that sshd will not
# let you log in afterall. Oh well...
die "target= not given on kernel command line, fatal, rebooting in 10mn" 600
fi

[ -z "$TARGET" ] && TARGET="/grhat/targets"

if [ -z "$PROTOCOL" ]; then
# Little trick to make PROTOCOL default to the one used to retrieve the
# kickstart image (which doesn't mean you can't kickstart from http and
# install from NFS; just set PROTOCOL in that case)
PROTOCOL=`cat $CMDLINE $CMDLINE2 | grep "ks=" | tail -1 | sed -e "s/.*ks="//`
if [ -z "$PROTOCOL" ]; then
  log "LOGWARNING: Couldn't read protocol from cmdline, defaulting to http"
  PROTOCOL=http
else
  log "Assumed PROTOCOL $PROTOCOL from ks kernel command line"
fi

if [ "$PROTOCOL" != nfs -a "$PROTOCOL" != http ]; then
  die "Fatal: No support for protocol $PROTOCOL, rebooting in 10mn" 600
fi

# This one, we want in the config file no matter what (we won't write
# all the other values if they are default)
echo "PROTOCOL=$PROTOCOL" >> /etc/sysconfig/getupdates
fi
```

```
screen
Eterm Font Background Terminal

# If you do multiple runs during debugging, those are already present
# and immutable
chattr -i /var/lib/grhat/funcs /usr/local/scripts/getupdates /usr/local/bin/shlock
ck 2>/dev/null
if [ "$PROTOCOL" = http ]; then
  if [ -z "$HTTUPDPATH" ]; then
    HTTUPDPATH="http://apt/"
    log "httpupdpth unset in kernel command line"
    log "defaulting to $HTTUPDPATH"
  fi
  BASETREE="$HTTUPDPATH:$TARGET:$TARGET/links/"
  KERMODTREE="$HTTUPDPATH:$TARGET/./ks.cfg/modules/`uname -r`/"
  log "Got httpupdpth from kernel command line"
  log "Retreiving updates from $BASETREE"
  pushd /tmp
  file=shlock; wget $BASETREE/$file; mv $file /usr/local/bin
  file=getupdates; wget $BASETREE/$file; mv $file /usr/local/scripts
  file=funcs; wget $BASETREE/$file; mv $file /var/lib/grhat
  popd

# The RH installer should do that for us, but I'll leave this here
# just in case
if ! /sbin/lsmod | grep -q nfs; then
  log "Trying to load NFS support for getupdates install logging"
  file=sunrpc.o; wget $KERMODTREE/$file; /sbin/insmod -f $file
  file=lockd.o; wget $KERMODTREE/$file; /sbin/insmod -f $file
  file=nfs.o; wget $KERMODTREE/$file; /sbin/insmod -f $file
else
  log "NFS support loaded, getupdates can use that"
fi
else
if [ -z "$NFSSHST" -o -z "$NFSSUPDPATH" ]; then
  NFSSHST=software.nfs
  NFSSUPDPATH=/software
  log "nfshost and/or nfssupath unset in kernel command line"
  log "defaulting to NFS from $NFSSHST:$NFSSUPDPATH"
fi
  BASETREE=/auto/software:$TARGET:$TARGET/links/
  log "Retreiving updates from $BASETREE via NFS"

  mount -o ro,nolock $NFSSHST:$NFSSUPDPATH /auto/software
  cp $BASETREE/shlock /usr/local/bin
  cp $BASETREE/funcs /var/lib/grhat/funcs
  cp $BASETREE/getupdates /usr/local/scripts
fi

chmod 755 /var/lib/grhat/funcs /usr/local/scripts/getupdates /usr/local/bin/shlock
ck
# This is for getupdates to know the files came from us, not RH
chattr +i /var/lib/grhat/funcs /usr/local/scripts/getupdates /usr/local/bin/shlock
if ! /usr/local/scripts/getupdates --firstload; then
  die "getupdates exited with error code, rebooting in 10mn" 600
fi
echo

test -d /auto/software && umount /auto/software
log "=== Post Install Done ==="
```

# Implementation: getupdates funcs

- All changesets have to include funcs to get wrappers and install functions:
  - reads \$WORKDIR / \$LEADGOLD / \$MACHTYPE (srv vs wks)
  - installlink: move a file out and make a symlink (like resolv.conf)
  - installfile: complex function that provides 8 ways to install a file
  - chatr: only run if argument is file, and skip symlinks
  - removefiles: delete file only if it's there, and after chatr -i
  - rotatefile: move a file out before a new one is installed (used by changesets and installfile, before replacing a stock file)
  - rpm/dpkg wrapper that removes expected stderr output
  - apt-get wrapper that defaults to -q --trivial-only for safety
  - apt-get-force when you really need it (apt-get -q -y)

# Implementation: installfile

- We need to take care of many cases:
  - if file is mutable, rotate out, install new file as immutable, and record operation in WORKDIR/state/changedfiles
  - optionally, install without rotation (overwrite), like in /etc/cron.daily
  - if file is mutable but in changedfiles, it's been user modified, skip
  - if file is immutable, upgrade getupdates maintained file in place
  - or force rotation anyway
  - or install a file in place and leave mutable by specifying the md5sum of the intended target (which leaves it alone if the user modified it)

# Implementation: Changeset Rules

- There are many rules for writing changesets:
  - changesets are numbered in the order you need them to run
  - changesets can be run multiple times and have to be written accordingly
  - do not generate any stderr output unless it's a warning or error
  - changesets run under bash's errexit. All lines of code have to return true
  - you cannot change the current directory, but you can use pushd/popd
  - you must not use apt-get install -y foo to get its dependencies: list all the dependencies and remove conflicting packages first. This safeguards you against some package removing other ones you didn't expect
  - remember, this all runs unattended, expect the unexpected and be very careful



# Implementation: Changeset Push

- copy changeset code to targets/wkslead/xxxxx/runme
- note that new installs could break at that time, test on lead, not on gold
- copy new files used by changeset in xxxx/files/....
- use makefile to rebuild workfiles.tar.bz2
- go on a client and remove \$WORKDIR/state/xxxx
- run getupdates to check changeset on a client
- change updatedate on the server for other clients to pick up the new changeset

# Implementation: apt trees

- apt tree setup obviously distro specific
- apt-rpm setup example available with getupdates distro
- recommended setup:
  - aptroot/targets/lead
  - aptroot/targets/gold
  - aptroot/targets/links
- makefile to sync from main apt tree to lead and then gold
- optional extras shadowed out with pinning

# Stats and logs: clientinfo

- Keep track of how many machines you have, what they run and whether updates are failing
  - /auto/clientinfo: hostname -> MAC and IP -> MAC
  - /auto/clientinfo/MAC (multiple versions in ontap snapshots)
    - ◆ hardware config info: cpuinfo, df, lspci
    - ◆ system info: ps-auxww, free, uname-a
    - ◆ machine info in symlinks: hostname, ip, uuid (RHN)
    - ◆ getupdates run info: log/log.last/log.update
    - ◆ machine load info: target (wkslead vs srvgold...), rpm-qa
    - ◆ user info: owner, who

## pushing: runcmd

- Uses ssh key root logins to send commands to all machines
- Imperfect: any client can be up or down at any time
- Uses:
  - Mostly recovering from complete getupdates failure
  - Dirty fix on “most” machines
  - trigger a pull on most machines (runcmd getupdates)
  - quick info poll without writing a changeset

# Thank god for symlinks

- /etc/resolv.conf -> per office subdomains for service names
- /etc/crontab -> crontab.srv | wks
- /etc/auto.net
- /etc/syslog.conf
- /etc/nsswitch.conf (with/without ldap/nis)
- /etc/pam.d/system-auth (with/without kerberos/ldap)
- /etc/X11/XF86Config
- etc..

# What's missing / TODO

- Automated, fully failsafe, rollbacks
  - quite hard to implement without snapshots in the filesystem (lvm2 ?)
- Add checksums/gpg signatures
  - not too hard to do, but mostly useless if you don't make sure all your contrib rpms/debs are also signed -> lots of work
- crond watcher and restarter
  - Need to get around to that one day :)
- clientinfo reporting over HTTP and not just NFS
  - would be a little work and require a specialized CGI with http upload
- backup getupdates when you sync a bad getupdates
  - critical missing piece if you update getupdates and make a fatal mistake
  - watch /var/log/getupdates and retrieve and run shell script from a central server if the last update is too old (try 4 bytes of the IP, and then 3, 2, 1, and just failsafe)

# Questions