Linux.conf.au 2004

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

http://marc.merlins.org/linux/talks/getupdates/

Marc MERLIN
marc_soft@merlins.org

I have extra ice cream. Whatever shall I do ???

- I like ice cream
- but they only gave me 4
- I surely need more
- give me your ice cream

- My name is Russell Coker
- Give me your ice cream !!!

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 a comprized of a shell script and a compressed tar archive containing optional files used by the update (just two files to retreive, 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

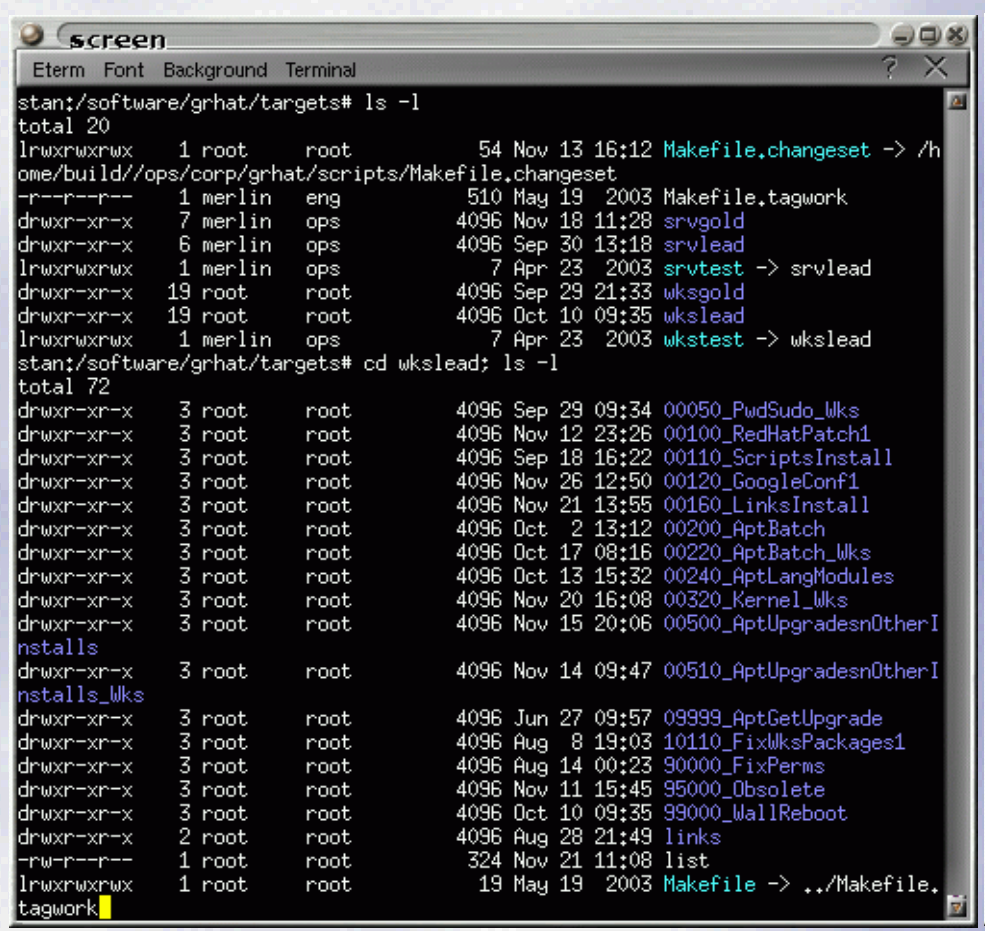
- /etc/sysconfig/getupdates:

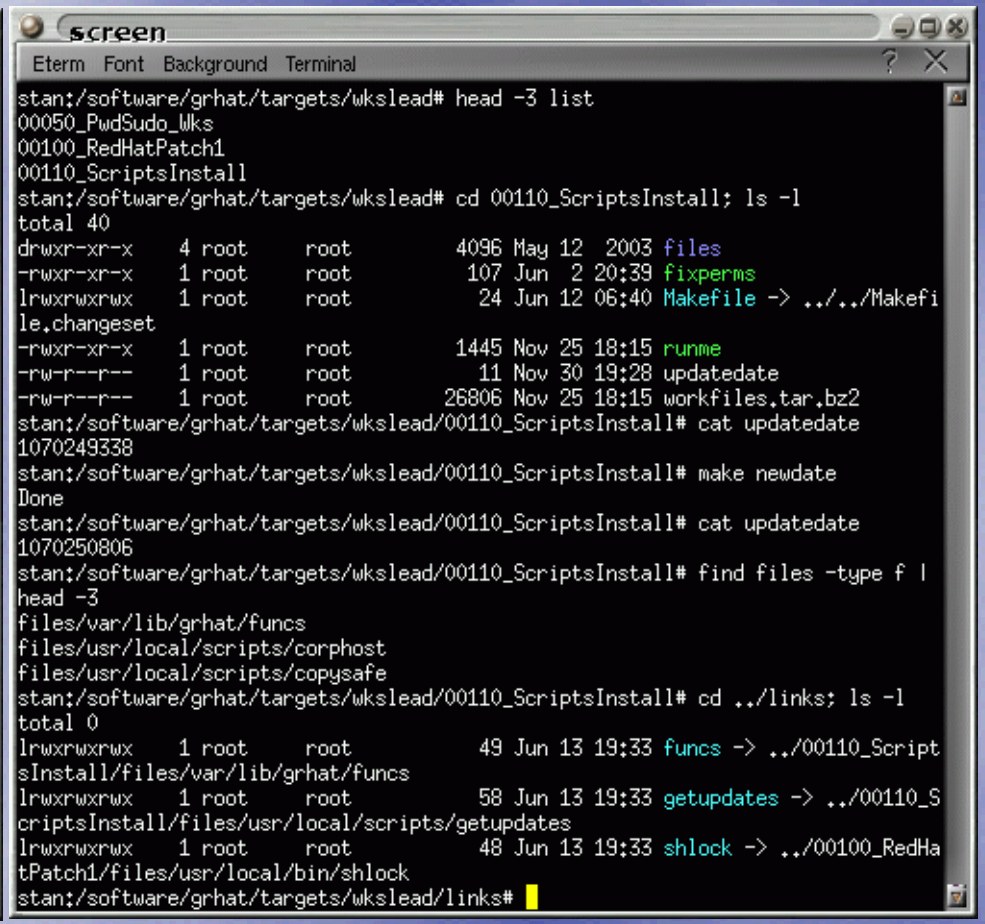
```
TARGET=wkslead
NFSHOST=software.nfs
NFSSWPATH=/software  -> $NFSHOST/$NFSSWPATH (boot media) or $NFSSWPATH (autofs)
PROTOCOL=nfs
WORKDIR=/var/lib/grhat
STATBASE=/auto/clientinfo -> runtime (autofs)
NFSSTATMNT=clientinfo.nfs:/clientinfo -> load time from boot media
```

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

Implementation: Layout

- File layout in the install tree:





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 retreive 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 $NFSHOST:/$NFSUPDPATH/$TARGD/
# HTTP loads are made from http://$HTTPUPDPATH/$TARGD/
# PROTOCOL defaults to http but you can set to nfs
for VAR in TARGET NFSHOST NFSUPDPATH HTTPUPDPATH TARGD PROTOCOL
do
    var=`echo $VAR | sed 's/\(.*\)/\L\1/'`
    VALUE=`cat $CMDLINE $CMDLINE2 | grep "$var=." | tail -1 | sed -e "s/.*$var=/
/" -e "s/ .*//" -e "s/\!\!/$DOMAIN/"`
    if [ z"$VALUE" != z ]; then
        eval `echo "$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 "$TARGD" ] && TARGD="/grhat/targets"

if [ -z "${PROTOCOL}" ]; then
    # Little trick to make PROTOCOL default to the one used to retreive 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 $CMDLINE2 | grep 'ks=.' | tail -1 | sed -e "s/.*ks=//
" -e "s/:.*//"`
    if [ -z "${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
```

52,1 45%

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/shlo
ck 2>/dev/null
if [ "$PROTOCOL" = http ]; then
    if [ -z "$HTTPUPDPATH" ]; then
        HTTPUPDPATH="http://apt/"
        log "httpupdpath unset in kernel command line"
        log "defaulting to $HTTPUPDPATH"
    fi
    BASETREE="$HTTPUPDPATH/$TARGD/$TARGET/links/"
    KERMODTREE="$HTTPUPDPATH/$TARGD/../ks.cfg/modules/`uname -r`/"
    log "Got httpupdpath 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 $file
    else
        log "NFS support loaded, getupdates can use that"
    fi
else
    if [ -z "$NFSHOST" -o -z "$NFSUPDPATH" ]; then
        NFSHOST=software.nfs
        NFSUPDPATH=/software
        log "nfshost and/or nfsswpath unset in kernel command line"
        log "defaulting to NFS from $NFSHOST:$NFSUPDPATH"
    fi
    BASETREE=/auto/software/$TARGD/$TARGET/links/
    log "Retreiving updates from $BASETREE via NFS"

    mount -o ro,nolock $NFSHOST:$NFSUPDPATH /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/shlo
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/shlo
ck
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 ==="
```

59,0-1 20%

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
  - chattr: only run if argument is file, and skip symlinks
  - removefiles: delete file only if it's there, and after chattr -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 Examples

Eterm Font Background Terminal

```
#!/bin/bash -e
# $Id: //depot/ops/corp/grhat/updates/00050_PwdSudo_Wks#7 $

. /var/lib/grhat/funcs

# Very important: this script could be run multiple times, you need to take this
# into account so that it doesn't do stupid things if called more than once

log "Starting $1"

# If you change this, change
# //depot/ops/corp/grhat/ks.cfg/snippets/ks_12_pwd_wks.cfg too
ENC='$1$XXXXXXXX$YYYYYYYYYYYYYYYYYYYYY'

if ! grep -q  ^sashroot: /etc/passwd; then
    echo "Installing sash account"
    useradd -d /root -M -s /sbin/sash -o -u 0 -g 0 sashroot
fi

# UPDATEPWD can bet set to no in /etc/sysconfig/getupdates
if [ ! z"$UPDATEPWD" = zno ]; then
    echo "Updating root password"
    perl -p -i -e "\$enc='$ENC'; chomp(\$enc); s!^root:[^:]*:!root:\$enc:!; s!^s
ashroot:[^:]*:!sashroot:\$enc:!" /etc/shadow
fi

if ! grep -q '^%sysops' /etc/sudoers; then
    echo "Installing sysops group sudo access"
    perl -p -i -e 's/^(root.*ALL=.*)/$1\n%sysops    ALL=(ALL) ALL/' /etc/sudoers
fi

log "Finished $1"
# Return success
exit 0
-- INSERT --                                              34,1          All
```

Eterm Font Background Terminal

```
# $Id: //depot/ops/corp/grhat/updates/00100_RedHatPatch1#46 $

. /var/lib/grhat/funcs

log "Starting $1"

echo "Basic RH fixes"
perl -p -i -e 's/1:2345:respawn:\/sbin\/mingetty tty1/1:2345:respawn:\/sbin\/min
getty --noclear tty1/' /etc/inittab
# Of course we want sysrq. How can you fix an unrebootable machine otherwise?
perl -p -i -e 's/kernel.sysrq = 0/kernel.sysrq = 1/' /etc/sysctl.conf

echo "Installing new programs in /usr/local/"
installfile usr/local/scripts copysafe
installfile usr/local/scripts savelog

echo "Installing Broadcom NIC driver"
if ! rpm -qa | grep "^bcm5700-" &>/dev/null; then
    rpm -i http://apt/RedHat/grhat/links/bcm5700-2.4.20-8.rpm &>/dev/null || tru
e
    rpm -i http://apt/RedHat/grhat/links/bcm5700-2.4.20-8-smp.rpm &>/dev/null ||
 true
fi

echo "Installing new config files"
installfile etc bashrc
installfile etc zshrc
installfile etc inputrc

# RH zsh is borked, it doesn't source /etc/profile and provides a bad zprofile
(cd /etc; ln -snf profile zprofile )

installfile etc/mc '*'
installlink . /etc/mc/mc.lib /usr/share/mc/mc.lib
installlink . /etc/mc/mc.ini /usr/share/mc/mc.ini
installlink . /etc/mc/mc.ext /usr/share/mc/mc.ext

if [ ! -L /etc/alternatives/etags -a -x /usr/bin/etags -a ! -x /usr/bin/etags.em
acs -a -x /usr/bin/ctags ]; then
    mv /usr/bin/etags{,.emacs}
fi

if [ ! -x /usr/bin/etags -a -x /usr/bin/ctags ]; then
    ln -snf /usr/bin/ctags /etc/alternatives/etags
    ln -snf /etc/alternatives/etags /usr/bin/etags
fi

# Quiet down stupid, spammy cron job
TARGET=/etc/cron.d/sysstat
if test -f $TARGET && ! grep -q '/dev/null' $TARGET; then
    perl -p -i -e 's#(/usr/lib/sa.*)#$1 &>/dev/null#' $TARGET
fi

echo "Installing cron jobs"
# All cron jobs need to be installed with -o to avoid rotations (otherwise
# we get multiple crons, not something you want)
installfile -o etc/cron.d getupdates.cron
installfile var/lib/grhat knowncronerrors

echo "Removing RH programs"
# We call /bin/rpm to get the return code from rpm (and not call the function)
for rpm in firstboot rhn-applet up2date-gnome up2date desktop-printing
do
    if /bin/rpm -q $rpm >/dev/null; then
        echo "Removing $rpm"; rpm --erase $rpm
    fi
done

log "Finished $1"
# Return success
exit 0
                                                       68,6          Bot
```

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 (mutiple 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)
- Switch to YUM instead of apt-get on Red Hat?

Are you a good sysadmin,
or programmer?

Need a job?
(in the Silicon Valley or New York)

Email: marc_jo@merlins.org

Questions