Updates of Multiple Machines Using SSH

Šarūnas Burdulis
September 7, 2007

Inspired by [1] here is a technique for updating multiple hosts on a network using Secure Shell (SSH), private/public keys, `sudo` and some scripting. Though this assumes Linux machines built using Debian, Ubuntu or other APT/dpkg distributions, it can also be applied to any OS, for which command line software update tools, SSH, and sudo (or their equivalents) exist.

Key elements are:

- SSH daemons running on every host;
- user account on each of the hosts with appropriate sudo privileges;
- account on one of the hosts or on a separate administrative workstation with its public key distributed to all the other hosts.

1 SSH

We start on administrator's workstation where a regular user's SSH key pair is generated (`ssh-keygen`) with no passphrase for the private key. As a result `.ssh/` should contain `id_dsa` and `id_dsa.pub` files with private and public keys respectively. The latter should be distributed to all the hosts, which we intend to update via SSH:

```
bofh@admins$ scp .ssh/id_dsa.pub bofh@host:~/.ssh/authorized_keys
```

This assumes that account `bofh`: 1) exists on all the hosts; 2) has `.ssh/` directory already. As a result user `bofh` should be able to login via SSH to any of the hosts without being prompted for the password.

2 `sudo`

Software updates usually require super-user privileges. As we do not allow SSH logins as `root`, user `bofh` has to be given appropriate rights. By adding the following line to `/etc/sudoers`

```
bofh  ALL=NOPASSWD: /usr/bin/apt-get
```

user `bofh` is allowed to run `apt-get` via `sudo` without being prompted for their password, for example:

```
bofh@host$ sudo apt-get update
```

or via SSH from `admins` workstation:

```
bofh@admins$ ssh bofh@host sudo apt-get update
```

3 Script Sample

The script given below can be used to streamline updates of multiple hosts. There are of course countless ways to adapt this to your needs.

```
#!/bin/bash
# ~/bin/update -- script to update groups of Debian/Ubuntu machines
#
# Usage: update <name> [<name> [<name>]]
#
# where <name> is a nickname for predefined class of machines:
# serv --- main servers
# grid --- computational grid nodes
# wkst --- workstations

usage()
{
   printf "\n%s\n\n" " Usage: update < serv | grid | wkst >"
   exit 1
}

if [ -z $1 ] ; then
   usage
fi

TIMEOUT=10

COMMAND="sudo /usr/bin/apt-get update && sudo /usr/bin/apt-get upgrade"

SERV="host1 host2 host3"
GRID="node1 node2 node3 node4 node5"
WKST="etch1 etch2 etch3 gutsy1 gutsy2 gutsy3"

while WHAT=$1 ; shift ; do

   case $WHAT in
       serv) MACHINES=$SERV;;
       grid) MACHINES=$GRID;;
       wkst) MACHINES=$WKST;;
       *) usage;;
   esac

   for machine in $MACHINES
   do
       printf "\n\n%s\n\n" $machine:
       ssh -oConnectTimeout=$TIMEOUT -t $machine $COMMAND
   done

done
```

NB: `apt-proxy`

Using `apt-proxy` (`aptcacher`, other?) on a local network only makes sense while updating a number of similarly configured machines.

References

[1] Martin Brown, "System Administration Toolkit: Distributed administration using SSH" (http://www.ibm.com/developerworks/aix/library/au-satdistadmin/index.html?ca=drs-tp3307)