

Keeping in Sync:

Secure use of ssh and rsync with cron

Alex Owen - QMUL

r.a.owen@qmul.ac.uk

The Problem

- Need to repeatedly and automatically copy files from one server to another. Possibly across administrative domains.
- My use case: keeping dnsmasq config in sync on a 2 node master/slave cluster

The Tools

- cron - again and again and again and again...
- ssh - password-less auth (keys) and secure
- rsync - efficient file sync
 - useful server mode

SSH: Secure use of keys

- Passphrase-less key needed for cron so:
 - => bound to single command (or script)
 - => single source IP

`/etc/ssh/sshd_config`

`PermitRootLogin forced-commands-only`

`/root/.ssh/authorized_keys`

(on one line prepended to the key)

`from="192.0.2.1",command="/sbin/shutdown -r now",`

`no-port-forwarding,no-X11-forwarding,no-agent-forwarding`

RSYNC: server mode

- `rsync` can run as a server
 - => config file defines what is shared
 - => config file defines read-only/read-write

`/etc/rsync.conf`

`[etc]`

```
path = /etc
comment = /etc
uid = root
gid = root
read only = yes
```

```
exclude =/***
```

```
include = /hosts /ethers
```

- client uses “`::`” to access module on the server

```
rsync server::etc/hosts ./
```

Putting it together

- Bind ssh-key to an rsync server process that has its own private config file

`/root/.ssh/authorized_keys`
(on one line prepended to the key)

```
from= "192.0.2.1",  
command="/usr/bin/rsync --server --daemon  
--config=/etc/my-rsyncd.conf",  
no-port-forwarding,no-X11-forwarding,no-agent-forwarding
```

Running with cron

```
rsync $RSYNCOPTS -e ssh server::etc/hosts ./
```

```
#RSYNC OPTIONS
#-v = verbose
#-x = one filesystem
#-a = -rlptgoD
# -r = recursive
# -l = copy symlinks as symlinks
# -p = preserve permissions
# -t = preserve modification times
# -g = preserve group
# -o = preserve owner (super-user only)
# -D = preserve device and special files
#-H = preserve hard links
#-W = copy files whole (w/o delta-xfer algorithm)
#-A = preserve ACLs
#-X = preserve extended attributes

RSYNCOPTS="-xaHWAX" # See above
```

Other rsync options

- `-S == --sparse` : handle sparse files efficiently
- `-n == --dry-run` : use this to test
- `-i == --itemize-changes`

```
getetcfile(){
local filename
local output
filename="$1"

output=$(rsync $RSYNCOPTS -i -e ssh $SERVER::etc/${filename} /etc/${filename}
2>/dev/null)

if [ -z "$output" ] ; then
    return 1
else
    return 0
fi
}

#Usage:
if getetcfile hosts; then
    DNSMASQRELOAD=true
fi
```

Script tips: Lockfile

```
#!/bin/bash
LOCKDIR=/var/run
LOCKFILE=${LOCKDIR}/my.lock

#Setup functions
removelock(){
    rm -f $LOCKFILE
    trap : EXIT
    GOTLOCK=0
}

trytolock(){
    ln -s $HOSTNAME-$$ $LOCKFILE 2>/dev/null || return 2
    trap removelock EXIT
}

#main()
trytolock || exit 1

#do stuff

remove lock
exit 0
```

Summary

- sshd only allow root if bound command
- Bind ssh key to command from single IP
- Bind to rsync server w/ custom config
- configure rsync modules to be as restrictive as you like
- use rsync '::' syntax and module paths in cron job