Full credit is given to the above companies including the OS that this PDF file was generated!

Rocky Enterprise Linux 9.2 Manual Pages on command 'ssh-copy-id.1'

$ man ssh-copy-id.1

SSH-COPY-ID(1)   BSD General Commands Manual   SSH-COPY-ID(1)

NAME

ssh-copy-id ? use locally available keys to authorise logins on a remote
machine

SYNOPSIS

ssh-copy-id [-f] [-n] [-s] [-i [identity_file]] [-p port] [-o ssh_option]
[user@]hostname

ssh-copy-id -h | -?

DESCRIPTION

ssh-copy-id is a script that uses ssh(1) to log into a remote machine
(presumably using a login password, so password authentication should be
enabled, unless you've done some clever use of multiple identities). It
assembles a list of one or more fingerprints (as described below) and
tries to log in with each key, to see if any of them are already in?
stalled (of course, if you are not using ssh-agent(1) this may result in
you being repeatedly prompted for pass-phrases). It then assembles a
list of those that failed to log in, and using ssh, enables logins with
those keys on the remote server. By default it adds the keys by append?

ing them to the remote user's ~/.ssh/authorized_keys (creating the file, and directory, if necessary). It is also capable of detecting if the re?mote system is a NetScreen, and using its ?set ssh pka-dsa key ...? com?mand instead.

The options are as follows:

-i identity_file

Use only the key(s) contained in identity_file (rather than look?ing for identities via ssh-add(1) or in the default_ID_file). If the filename does not end in .pub this is added. If the filename is omitted, the default_ID_file is used.

Note that this can be used to ensure that the keys copied have the comment one prefers and/or extra options applied, by ensuring that the key file has these set as preferred before the copy is attempted.

-f   Forced mode: doesn't check if the keys are present on the remote server. This means that it does not need the private key. Of course, this can result in more than one copy of the key being installed on the remote system.

-n   do a dry-run. Instead of installing keys on the remote system simply prints the key(s) that would have been installed.

-s   SFTP mode: usually the public keys are installed by executing commands on the remote side. With this option the user's ~/.ssh/authorized_keys file will be downloaded, modified locally and uploaded with sftp. This option is useful if the server has restrictions on commands which can be used on the remote side.

-h, -? Print Usage summary

-p port, -o ssh_option

These two options are simply passed through untouched, along with their argument, to allow one to set the port or other ssh(1) op?tions, respectively.

Rather than specifying these as command line options, it is often better to use (per-host) settings in ssh(1)'s configuration file: ssh_config(5).

Default behaviour without -i, is to check if ?ssh-add -L? provides any output, and if so those keys are used. Note that this results in the comment on the key being the filename that was given to ssh-add(1) when the key was loaded into your ssh-agent(1) rather than the comment con?tained in that file, which is a bit of a shame. Otherwise, if ssh-add(1) provides no keys contents of the default_ID_file will be used.

The default_ID_file is the most recent file that matches: ~/.ssh/id*.pub, (excluding those that match ~/.ssh/*-cert.pub) so if you create a key that is not the one you want ssh-copy-id to use, just use touch(1) on your preferred key's .pub file to reinstate it as the most recent.

EXAMPLES

If you have already installed keys from one system on a lot of remote hosts, and you then create a new key, on a new client machine, say, it can be difficult to keep track of which systems on which you've installed the new key. One way of dealing with this is to load both the new key and old key(s) into your ssh-agent(1). Load the new key first, without the -c option, then load one or more old keys into the agent, possibly by ssh-ing to the client machine that has that old key, using the -A option to allow agent forwarding:

```
    user@newclient$ ssh-add
    user@newclient$ ssh -A old.client
    user@oldl$ ssh-add -c
    ... prompt for pass-phrase ...
    user@old$ logoff
    user@newclient$ ssh someserver
```

now, if the new key is installed on the server, you'll be allowed in un?prompted, whereas if you only have the old key(s) enabled, you'll be asked for confirmation, which is your cue to log back out and run

```
    user@newclient$ ssh-copy-id -i someserver
```

The reason you might want to specify the -i option in this case is to en?sure that the comment on the installed key is the one from the .pub file, rather than just the filename that was loaded into your agent. It also ensures that only the id you intended is installed, rather than all the keys that you have in your ssh-agent(1). Of course, you can specify another id, or use the contents of the ssh-agent(1) as you prefer.

Having mentioned ssh-add(1)'s -c option, you might consider using this whenever using agent forwarding to avoid your key being hijacked, but it is much better to instead use ssh(1)'s ProxyCommand and -W option, to bounce through remote servers while always doing direct end-to-end authentication. This way the middle hop(s) don't get access to your ssh-agent(1). A web search for ?ssh proxycommand nc? should prove enlightening (N.B. the modern approach is to use the -W option, rather than nc(1)).

SEE ALSO

ssh(1), ssh-agent(1), sshd(8)

BSD June 17, 2010 BSD