If you use SSH a lot or if you use it in automated tools, you’ll no doubt become annoyed  
by the need to type a password with every connection. There is a way around this requirement: You can set up the SSH client with keys and give the client’s public key to the server  
computer. With this configuration, the SSH client computer can identify itself, possibly  
obviating the need for you to type a password.

Configuring SSH to operate without the use of passwords is convenient,  
but it does increase security risks. If somebody you don’t trust ever gains  
access to your account on the SSH client system, that person will be  
able to log into the SSH server system as you without the benefit of your  
password. Thus, you should create a password-less login only from a  
client that’s very well protected, if at all. Configuring access to the root  
account in this way is particularly risky!

To configure SSH not to require a password, follow these steps:  
**1.** Log into the SSH client system as the user who will be performing remote access.  
**2.** Type the following command to generate a version 2 SSH key:  
$ **ssh-keygen -q -t rsa -f ~/.ssh/id\_rsa -C '' -N ''**

Step 2 generates a version 2 key. You can instead generate a version 1 key  
by typing **ssh-keygen -q -t dsa -f ~/.ssh/id\_dsa -C '' -N ''**. This  
generates id\_dsa and id\_dsa.pub files. This procedure is *not* recommended because SSH version 1 is not as secure as version 2.

**3.** Step 2 generates two files: id\_rsa and id\_rsa.pub. Transfer id\_rsa.pub to  
the SSH server computer in any way that’s convenient—via a USB flash drive, by using  
scp, or by any other means. Copy the file under a temporary name, such as temp.rsa.  
**4.** Log into the SSH server system. If you use SSH, you’ll need to type your password.  
**5.** Add the contents of the file that you’ve just transferred to the end of the ~/.ssh/  
authorized\_keys file. (This file is sometimes called ~/.ssh/authorized\_keys2, so you  
should check to see which is present. If neither is present, you may need to experiment.)  
Typing **cat ~/temp.rsa >> ~/.ssh/authorized\_keys** should do this job, if you  
stored the original file as ~/temp.rsa.  
**6.** On some systems, you may need to modify permissions on the ~/.ssh/authorized\_  
keys file and on the directories leading to it. The authorized\_keys file may require  
0600 permissions, and you may need to remove write permissions for any but the  
account’s owner on your home directory and on the ~/.ssh directory.

If you now log out of the SSH server system and try to log in again via SSH from  
the client, you shouldn’t be prompted for a password; the two computers handle the  
authentication automatically.  
If this doesn’t work, chances are the ~/.ssh/authorized\_keys file needs another name,  
as described earlier. You may also want to check that the file includes a line matching the  
contents of the original public-key file on the client. Some older clients may require you to  
specify that you use version 2 of the SSH protocol by including the -2 option:  
$ **ssh -2 server**

**Using *ssh-agent***  
Another SSH authentication option is to use the ssh-agent program. This program  
requires a password to initiate connections, so it’s more secure than configuring logins  
without passwords; however, ssh-agent remembers your password, so you need to type it  
only once per local session. To use ssh-agent, follow these steps:  
**1.** Follow the procedure for enabling no-password logins described in “Configuring  
Logins without Passwords,” but with one change: Omit the -N '' option from the  
ssh-keygen command in step 2. You’ll be asked for a passphrase at this step. This  
passphrase will be your key for all SSH logins managed via ssh-agent.  
**2.** On the SSH client system, type **ssh-agent /bin/bash**. This launches ssh-agent,  
which in turn launches bash. You’ll use this bash session for subsequent SSH logins.  
**3.** In your new shell, type **ssh-add ~/.ssh/id\_rsa**. This adds your RSA key to the set  
that’s managed by ssh-agent. You’ll be asked to type your SSH passphrase at  
this time.  
From this point on, whenever you use SSH to connect to a remote system to which  
you’ve given your public key, you won’t need to type a password. You *will*, however, have  
to repeat steps 2 and 3 whenever you log out, and the benefits will accrue only to the shell  
launched in step 2 or any shells that you launch from that one.  
If you make heavy use of this facility, you can insert ssh-agent into your normal login  
procedure. For instance, you can edit /etc/passwd so that ssh-agent /bin/bash is your  
login shell. For a GUI login, you can rename your normal GUI login script (for instance,  
change ~/.xsession to ~/.xsession-nossh) and create a new GUI login script that calls  
ssh-agent with the renamed script as its parameter. Either action inserts ssh-agent at the  
root of your user process tree so that any call to SSH uses ssh-agent.