Logging in with SSH keys

On this page:

- SSH Keys
    - Creating SSH keys (Linux and macOS)
    - Adding your SSH key to the ssh-agent (Linux and macOS)
        - Linux
        - macOS
    - Creating SSH keys (MobaXterm in Windows)
    - Copy Public Key
        - Method 1
        - Method 2

SSH Keys

SSH keys allow for a secure method of logging in to a server without the need of typing a password each time a connection is established. The process involves creating a key pair (one public and one private), which are essentially long strings of random numbers. Then, you place the public key on any remote server you wish to access (e.g. Magnus) and, when you want to login, your local SSH client presents the private key to the remote server. If the public and private key match, access is granted. Besides avoiding the need to type a password, SSH keys has several other advantages. For example, anyone monitoring your connection would not be able to intercept the password because it was never actually sent. Also, because SSH keys are based on public-key cryptography, they provide a more secure method of authentication than traditional passwords.

Below we have instructions on how to set up SSH keys on different operating systems.

Creating SSH keys (Linux and macOS)

Both Linux and MacOS come with a terminal application and SSH client already installed, so no additional programs are required.

On your local machine, open a terminal and run the command:

```
ssh-keygen -t rsa -f ~/.ssh/pawsey_rsa_key
```

This will create a new key pair named pawsey_rsa_key in your ~/.ssh directory, and will use the RSA algorithm for encryption. You will then be prompted to enter a passphrase, which is not your Pawsey password, but a passphrase to use for security of the generated key:

```
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): [Type a passphrase]
Enter same passphrase again: [Type the passphrase again]
```

We strongly recommend that you provide a passphrase for your key-pair. Should someone gain access to your private key they could log in to systems as you; a passphrase on your SSH key provides a secondary level of security. Note that your passphrase is only used to protect your SSH key, so it is never transmitted. Also, there are programs available (called SSH agents) that can securely manage your keys and passphrases and eliminate the need to type in your passphrase each time you log into a system (see below for more details).

Once you've entered your passphrase you will receive a confirmation:

```
Your identification has been saved in /home/<user>/.ssh/pawsey_rsa_key.
Your public key has been saved in /home/<user>/.ssh/pawsey_rsa_key.pub.
The key fingerprint is:
SHA256:0bCrfcR31Q5G9TN4TytTMsUAQI6chW9vi2b+cjbZE0M user@hostname
The key's randomart image is:
+---[RSA 2048]----+
|        ++....+o.|
|      ..=+   .o.o|
|       ++..  +o==|
|         *   E===|
|        S + oo..o|
|       o . + +o  |
|      . . + + o  |
|         * * o   |
|        +.=.. .  |
+----[SHA256]-----+
```

You should now have a public key (pawsey_rsa_key.pub) and a private key (pawsey_rsa_key):

```
ls ~/.ssh
pawsey_rsa_key
pawsey_rsa_key.pub
```

After generating the keys, you need to add them to the ssh-agent.

Adding your SSH key to the ssh-agent (Linux and macOS)

Linux

First, the ssh-agent needs to be started in the background. For that, you need to execute the following command within a terminal of your local machine:

```
eval "$(ssh-agent -s)"
```

which will respond an Agent pid.

In most flavours of linux, the key is added to the agent with a simple usage of the ssh-add command:

```
ssh-add ~/.ssh/pawsey_rsa_key
```

macOS

For macOS, steps are a bit different. First, start the ssh-agent in the background by executing the following command within a terminal of your local machine:

```
eval "$(ssh-agent -s)"
```

Second, you will need to modify your ~/.ssh/config file to automatically load keys into the ssh-agent and store passphrases in your keychain. Add the following lines to the ~/.ssh/config file:

```
Host *
 AddKeysToAgent yes
 UseKeychain yes
 IdentityFile ~/.ssh/pawsey_rsa_key
```

And finally, add the key to the ssh-agent by using the additional option -K (native to Apple's standard version of ssh-add):

```
ssh-add -K ~/.ssh/pawsey_rsa_key
```

Creating SSH keys (MobaXterm in Windows)

Make sure in the settings that your "Persistent home directory" is somewhere persistent, such as "_AppDataDir_\MobaXterm\home". Once this is done, start a local terminal window and follow the instructions above for creating a ssh key under Linux. You do not need to follow the instructions for ssh-agent, as MobaXterm does this automatically.

Remember that to be able to start a MobaXterm local terminal (since version 8), you need to download and install the CygUtils.plugin package from http://mobaxterm.mobatek.net/plugins.html. Move the downloaded file into the same directory as the MobaXterm executable, which is probably "C:\Program Files (x86)\Mobatek\MobaXterm Personal Edition", and then restart MobaXterm.

Copy Public Key

Once you have generated a key pair you need to add the public key to the server. Here are two methods to do it:

In the following examples, replace "**<user>**" with your Pawsey username, e.g. "**jsmith**@magnus.pawsey.org.au"

Method 1

```
ssh-copy-id -i ~/.ssh/pawsey_rsa_key.pub <user>@magnus.pawsey.org.au
```

Method 2

We will use a combination of SSH and the Linux command "cat" to paste the key to the server. You can use the following command:

```
cat ~/.ssh/pawsey_rsa_key.pub | ssh <user>@magnus.pawsey.org.au "mkdir -p ~/.ssh && cat >> ~/.ssh
/authorized_keys"
```

This command does the following:

- Prints the output of the local public key ~/.ssh/pawsey_rsa_key.pub
- Redirects the output to the Magnus login node (using your Pawsey username and SSH to log in)
- Creates a hidden directory in your home directory (~/.ssh) on Magnus
- Pastes the contents of the public key into the file ~/.ssh/authorized_keys, located on the Magnus login node

Note that /home is shared among all Pawsey systems, so this single command will set up SSH keys for each Pawsey system for which you have access. You do not need to repeat this for Zeus, Galaxy, hpc-data, etc.