SSH

Sterrewacht

Most of the desktop machine at the STRW can be accessed through the ssh protocol. So when you know your machine name, use that (including the strw.leidenuniv.nl domain) to access that machine directly.

If you do not have a personal machine you can use the `ssh.strw.leidenuniv.nl` virtual machine to log into our systems and continue from there with an ssh to any of the science servers or cluster machines.
Note that the %ssh.strw.leidenuniv.nl machine is just a gateway; it is not meant for any type of data processing, desktop environments etc. === Special access === Some places we visit (e.g. China or Iran) or some hotels abroad limit the internet access to web browsing only. Because you want more in such cases, the ssh server of the Sterrewacht now also serves the ssh protocol on web ports 80 and 443. So you can now get access to the Sterrewacht computer systems from those limiting environments using ssh ssh.strw.leidenuniv.nl -p 80 -l <your STRW accountname> With this type of connectivity you can add the tunnelling options (as indicated below) to gain connectivity to a windows remote desktop or your Linux VNC environment. ===== Instituut Lorentz ===== For Instituut Lorentz, the server is ssh.lorentz.leidenuniv.nl. Desktops and servers cannot be reached directly from outside, so you will always have to go through the ssh server first. But see also our [[institute_lorentz:institutelorentz_remoteaccess|list of details and tricks]] ===== LION ===== For LION, there is ssh3.physics.leidenuniv.nl. --------- ====== SSH tunnels ====== For the Mac and Linux commandline ssh client, setting up a tunnel is usually a matter of using the option \\ ''-L local_port:remote_machine:remote_port'', e.g. ''-L 3389:windows machine:3389'' \\ for forwarding a Windows remote desktop. More detail can be found in the [[vnc|vnc ssh tuning pages]]. See [[linux:putty]] for instructions about setting up a tunnel with ''putty%% (Windows, linux ssh client). ====== SSH client software ====== Linux and macOS come with a commandline client for ssh. For Windows, the recommended client is putty ====== SSH keys ====== ====Create a key pair==== To create an ssh key pair, with the proper encryption, open up a console on your local machine, and enter the following command: $ ssh-keygen -t ed25519 This results in the following output: Generating public/private ed25519 key pair. Enter file in which to save the key (/home/testuser1/.ssh/id_ed25519): Enter passphrase (empty for no passphrase): Enter same passphrase again: Your identification has been saved in /home/testuser1/.ssh/id_ed25519. Your public key has been saved in /home/testuser1/.ssh/id_ed25519.pub. The key fingerprint is: SHA256:gPD6FBuSJTpfkWCrpBPo7XoRqIEV+43g2sX2b6It2YI testuser1@ssh The key's randomart image is: +—[ED25519 256]—+ | .o*#*=. | | o..*+^o | | . .++E.* | |. .@.= . | | . ..X S | | . …= o | | . . o | | | | | +—-[SHA256]—–+ When asked for a “passphrase”, you should enter (a complex) one or optionally leave it blank. Note that without a passphrase your key pair will be free to use by anyone that has illegally gained access to your keys. MacOS and Linux also have a feature where keys are unlocked using your login password. The passphrase should be known to you only. **Keep your private key and passphrase as secret as you would keep your password!** The ssh-keygen program will now generate both your public and your private key. Your keys are stored in the .ssh/ directory in your home directory. The file id_ed25519 contains your

private key. YOU SHOULD GUARD THIS KEY WITH YOUR LIFE! This key is used to gain access on systems which have your private key listed in their authorized keys file. We cannot stress this enough, do not have your keys drifting around. Also, make sure your private key always is chmod 600, so other users on the system won't have access to it. The file id_ed25519.pub'' contains your public key, which can be added to other system's authorized keys files.

Simplified version in case of a shared home disk

This is how you authorize the key for use within a local network with shared home disk (so this is how to set up a key so you can log in using ssh without password between computers at the institute). See below for the general case of accessing a remote system.

Simply add the public part of the key to your .ssh/authorized_keys file, and make sure that that file is not accessible for others:

```
cat ~/.ssh/id_ed25519.pub >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
```

Copy public key to server

1. modern and easy, if it works

Nowadays, ssh comes with a utility to send a public key to a remote machine (requiring you to log in using your password once, or requiring a previous key to be already in place). This will take care adding the key to the authorized_keys on the remote system. To do this, simply use:

```
ssh-copy-id -i id_ed25519.pub user@remotehost
```

Actually, if you only have one key pair, you can leave out the -i and the name of the key to be copied, so this will do:

```
ssh-copy-id user@remotehost
```

2. the old way

To be able to log in to remote systems using your pair of keys, you will first have to add your public key on the remote server to the authorized_keys file in the .ssh/ directory in your home directory on the remote machine.

In our example we will assume you don't have any keys in the authorized_keys files on the remote server.

First we will upload the public keys to the remote server:

```
$ cd .ssh/
```

```
$ scp id_ed25519.pub user@remotehost:./id_ed25519.pub
id_ed25519.pub    100%
|**************************************************|   526       00:00
```

This will place your keys in your home directory on the remote server. After that we will login on the remote server using ssh the conventional way... with a password.

When you are logged in you should create a .ssh directory, and inside the .ssh/ directory create an authorized_keys file and add the keys to the file. Make sure the files are not readable for other users/groups. chmod 600 authorized_keys does the trick.

Placing the key works as follows:

```
$ cd .ssh
$ touch authorized_keys
$ chmod 600 authorized_keys
$ cat ../id_ed25519.pub >> authorized_keys
$ rm ../id_ed25519.pub
```

From now on you can login from client to server without having to specify a password (just a passphrase).