Login via SSH

Secure Shell

The SecureSHell

→ Client-Server-System for a secure (encrypted) connection and login to a remote host.

Info box

- Requirements: name of the remote machine (host name + domain name OR IP-address) as well as user name plus corresponding password.
- During the FIRST secure login to a new remote host, the "fingerprint" of the host is shown and the user is queried whether to trust that specific host machine: → Reply: yes
  (Unless of course you have reasons not to . . .)
- If you want to enforce use of SSH-login via password (e.g., because pubkey login is messed up):

```
ssh -o PreferredAuthentications=keyboard-interactive -o PubkeyAuthentication=no ...
```

SSH with Linux and macOS

The SSH-client is already installed for Linux/Unix and macOS (also for Win10, see further down).

1. Linux: open terminal/console;
   macOS: Applications → Utilities → Terminal
2. Login via Username (e.g. htodt) and Hostname (e.g. bell)

```
ssh htodt@bell.stud.physik.uni-potsdam.de -Y
```

   IMPORTANT! There is always at least once space character between a command and its options. Everything is case sensitive! The common option `-Y` activates X11-forwarding (graphics and window-widgets). An alternative to providing the hostname is the direct usage of the IP address, e.g. `ssh htodt@141.89.178.71 -Y`

3. Verify the success of the connection via the simple command `hostname` (should now yield e.g. "bell"). The application `xeyes` opens up a graphical element (the eponymous eyes) and thereby demonstrates the successful X11-forwarding
   (to terminate it, press CTRL + C or STRG + C)

MacOS users may have to set up the support for X11 first: `https://www.xquartz.org`

SSH with Windows I

The integrated PowerShell of Windows 10 behaves pretty much like Linux with regard to SSH connections. It comes with its own SSH-Client and a very similar syntax (without a X11 environment, however):

```
ssh username@weber.stud.physik.uni-potsdam.de
```

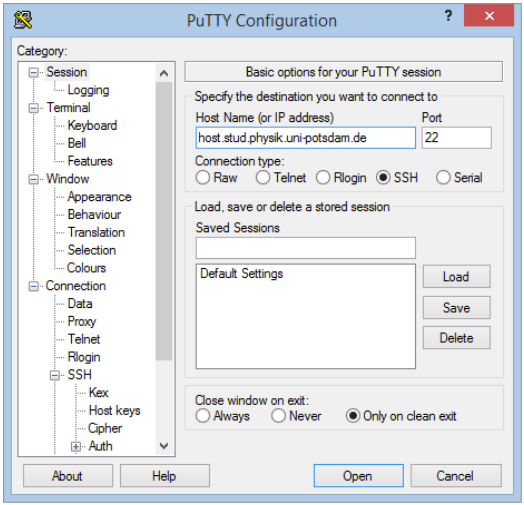
Older Windows versions (such as Window 8.1, etc) require the installation of a dedicated SSH-client, such as:

- PuTTY: `https://www.putty.org`
- MobaXterm: `http://mobaxterm.mobatek.net`
  $\rightarrow$ SSH-Client with X11-support, (recommended). Connect after launch with

```
ssh username@weber.stud.physik.uni-potsdam.de -Y
```

(use your username) to connect to a remote machine with X11-forwarding (in the given example line: weber). Password required.

PuTTY also allows for SSH-logins, no X11-forwarding for graphics, though.



1. Category → "Sessions"
2. Host Name† : Name or IP of a computer pool host, e.g. mahler.stud.physik.uni-potsdam.de Port: 22
3. → Open
4. Enter Username† and Password

†The Username can also be given directly in the command line before the `@` "Host Name": `user@host.stud.physik.uni-potsdam.de`

The SecureSHell after Login

After a successful login you execute commands ON THE host machine. The following commands should work for you:

`hostname`

→ shows (prints) the name of the host you are currently logged into. Call this also when you are unsure where you are currently typing on (local or remote machine).

`whoami`

→ shows your current user name (will likely differ from your own machine to your university pool account!)

`w`

→ shows user names of logged in users

`logout (or exit)`

→ end current SSH-session (not required, but nice)

Login without password I

By creating an RSA key pair and depositing the public key on the remote host one can login via SSH without the password:

1. In a terminal (Linux, macOS) or in the PowerShell (Windows) create a key pair using:

```
ssh-keygen
```

   You will be asked where to save the key pair and for a passphrase. Just confirm both default settings with ENTER (i.e. no passphrase).

2. Then change to the directory `.ssh` in your home directory (in doubt: `cd ; cd .ssh`).
3. From there copy your public key via `scp` to the the remote host:

```
scp id_rsa.pub user@host.stud.physik.uni-potsdam.de:.ssh/id_rsa.pub_laptop
```

   (assuming that there is already `.ssh` in your home directory on the remote host, if not, just execute `ssh-keygen` also once on the remote host)

Login without password II

4. Login via ssh on the remote host, change there to `.ssh` (is directly in your home directory) and *append* the previously copied public key `id_rsa.pub_laptop` to the file `authorized_keys`:

```
cat id_rsa.pub_laptop >> authorized_keys
```

Then, you should be able to login to the remote host without typing your password. Important note:

- By using `cat ... >> authorized_keys` you can append further public keys to the file without overwriting the key(s) already in that file
- The directory `.ssh` must always have the following permissions:
  `drwx------ 1 htodt astro 1112 Dec 9 15:48 .ssh`
  i.e. not readable for other users

The file .ssh/config for more comfort I

The file config in the directory .ssh allows, e.g., to shorten the host name of the remote host(s):

1. Create the file config in the directory .ssh with help of a *text editor*. Under Windows pay attention that the editor (e.g., Notepad) *does not* append the file extension .txt automatically. If necessary rename the file (e.g., mv config.txt config).
2. e.g., for the host bell and the user htodt create the following entry:

```
Host bell
Hostname bell.stud.physik.uni-potsdam.de
User htodt
```

3. subsequently, it is sufficient only to use the short name (here: bell), when starting a connection/tunnel with SSH in the terminal or PowerShell:

```
ssh bell
ssh -L 5903:localhost:5903 bell
```

The file `.ssh/config` for more comfort II

Remarks:

- You can create further entries for other remote hosts, insert a blank line for better readability between the entries for each host.
- Under Linux/macOS you can switch on automatic X11 forwarding (otherwise with the option `+Y`) with the following lines:

```
ForwardX11 yes
ForwardX11Trusted yes
```

- A frequent problem of SSH connections (at least in the past) is the interruption because of time-out. You may try to keep the connection "alive" by some kind of ping every 60 s with the following lines:

```
Host *
ServerAliveInterval 60
```

(The asterisk `*` means that the following line(s) apply to all remote hosts.)

The file .ssh/config for more comfort III

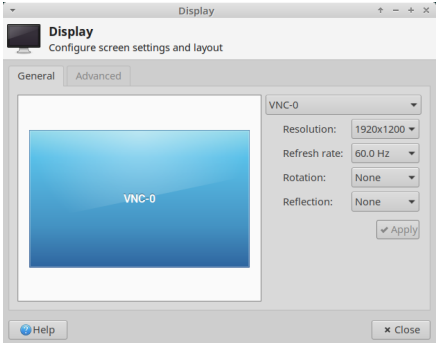
Example for a config file

```
Host *
ServerAliveInterval 60

Host bell
Hostname bell.stud.physik.uni-potsdam.de
User htodt
ForwardX11 yes
ForwardX11Trusted yes

Host joule
Hostname 141.89.178.77
User htodt
ForwardX11 yes
ForwardX11Trusted yes
```

Screen resolution in a running VNC session

Modern desktop environments (Xfce, LXQT, . . .) usually allow for changing the "screen resolution" of the running VNC session by the user.

e.g. under Xfce
→ openSUSE → Settings
→ Display
Pay attention that there is really the *VNC* display shown (here: VNC-0)!



→ this will also change the current size of the VNC session window

Alternatives: The `tiger` VNC viewer allows to change (permanently) the resolution by resizing the window of the viewer.
Also the command `xrandr` in your VNC session, e.g., `xrandr -s 1920x1080`, can set the resolution, but only to available resolutions.

Screen resolution of the VNC server

If your desktop environment doesn't support setting of the VNC-display resolution, you can alternatively also set a fixed resolution when starting the *VNC server*:

```
vncserver -geometry 1920x1200
```

$\rightarrow$ sets the initial default resolution to 1920$\times$1200 pixel (width $\times$ height) for the VNC session. Take into account that your own physical screen should have a sufficient resolution to display the VNC viewer window.

Links (clickable)

https://www.xquartz.org

https://www.putty.org

http://mobaxterm.mobatek.net

https://www.macports.org

https://www.java.com/de/download/manual.jsp