

PuTTY Tutorial

(Windows)

Table of Contents

PuTTY Client Basics and Simple Login.....	1
How to get PuTTY!.....	1
Simple Password-Based Login.....	1
Configuration Management.....	3
X-Forwarding.....	4
SSH Hopping.....	5
Copying Data.....	5
pscp.....	5
Filezilla.....	5
WinSCP.....	6
Public Key Authentication.....	6
Key Generation.....	7
Key Usage.....	8
Troubleshooting and Recovery.....	8
Summary.....	8
Port Forwarding / SSH Tunneling.....	9
Preparation.....	9
Tunnel and Connection to the same Server.....	10
Appendix.....	11
Useful/Entertaining Links.....	11

PuTTY Client Basics and Simple Login

How to get PuTTY!

Ask your administrator of your PC/Laptop/...!

If you are the administrator, go to the [PuTTY Web page](#) – or directly to the [Download page](#). There are installers (msi packages), which you can install as administrator. **Important:** Please check the GPG signatures of your download, and compare them with the Web page provided signatures!¹

If you are not administrator, and have not a good relation to your admin, there are also portable versions available (“Alternative binary files”). Those only need to be downloaded, and not installed. There is a zip archive, which contains the whole suite. Again! Check the GPG signatures!

Simple Password-Based Login

Once installed, or downloaded, you can start the PuTTY client by double-clicking it. You should see the following window.

¹ In order to guarantee security, you should be sure to use the original tools, and not a modified version from a Russian or Chinese hacker.

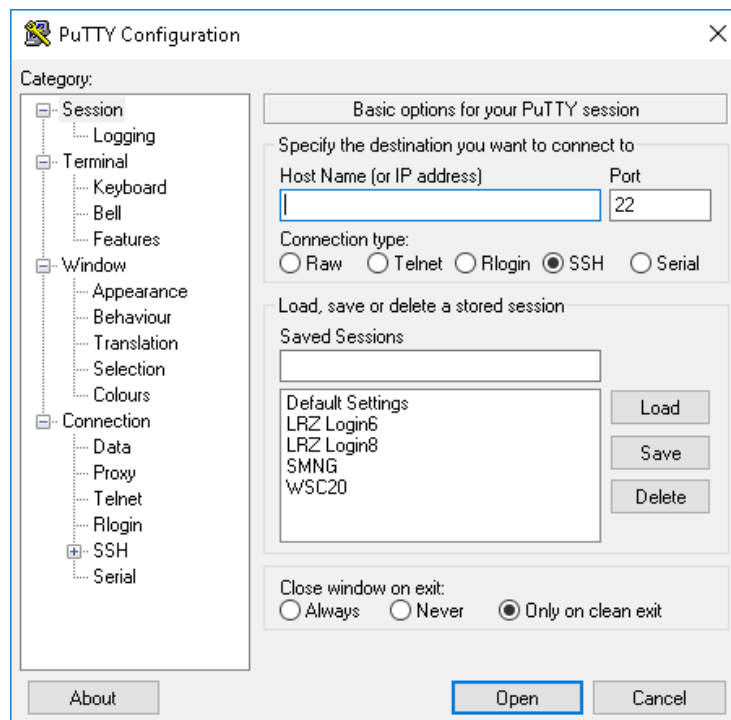
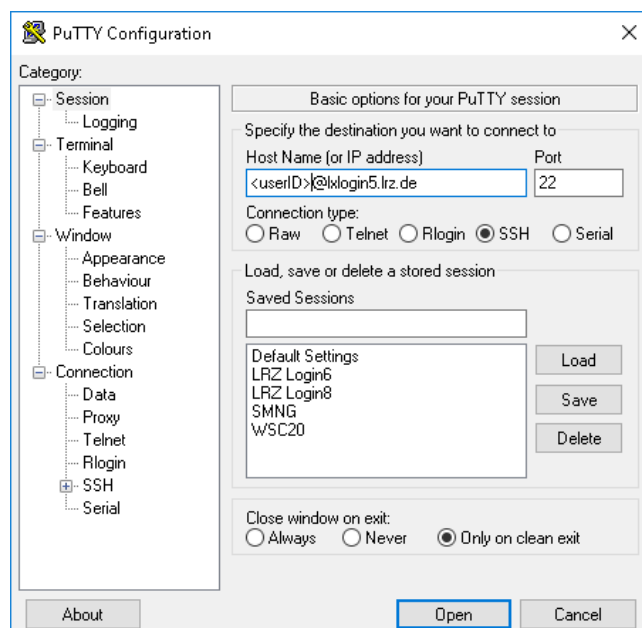


Figure 1: PuTTY Client GUI for configuration of SSH connections

Next, try to login! For the tutorial, the following login nodes are available:
lxlogin1.lrz.de, lxlogin2.lrz.de, lxlogin3.lrz.de,
lxlogin4.lrz.de, lxlogin8.lrz.de

Which one you pick, does not matter! You also got a User ID (we use <user ID> as placeholder; please replace it by your own ID!) and a corresponding password.

Try a simple Login like this:



After pressing Open, the GUI closes, and another window opens.

At the first login there you will probably be asked something like this:

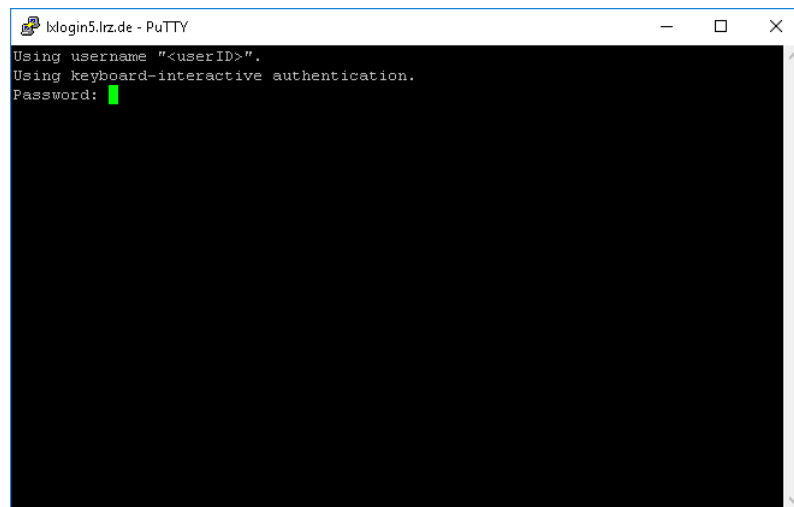
The authenticity of host 'lxlogin1 (129.187.20.105)' can't be established.

ECDSA key fingerprint is
SHA256:YmTuVciNdQzoZXpiDC4encMuUa8WIjJuA4NqmXaXgeM.

Are you sure you want to continue connecting (yes/no)?

Essentially, that's an authentication message from the remote server. You should not simply answer 'yes'! You should look the [LRZ Webpage](#), where these keys are trustworthy (the correct keys), in order to avoid [man-in-the-middle attacks](#). These key fingerprints are stored inside your local \$HOME/.ssh folder (~/.ssh/known_hosts).

Next, you are asked for your password. Enter it (the correct one)!



Now, you are logged in remotely, and have a bash shell open, where you can work in. For instance, to be check that you are really remotely working, issue:

```
> hostname
```

If you fail to login, call the course instructor!

On *success*², log out again! Either by

```
> exit
```

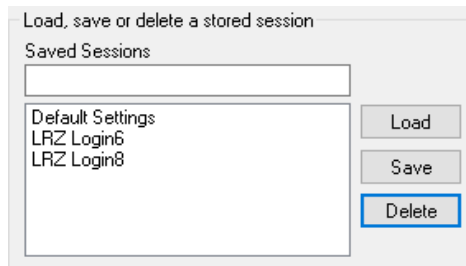
or by pressing 'Ctrl+D'. The window should close.

Configuration Management

The GUI provides a lot of features, you should learn to use immediately from the beginning. Because, if security were not easy and convenient to use, nobody would strive for security!

The first thing is that you can save and load session configurations. You find in the start GUI of PuTTY immediately the following part.

² `hostname` will show something like `cm2login1`. That's the internal name. We will use it later!

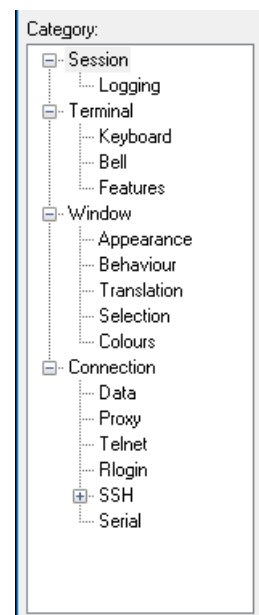
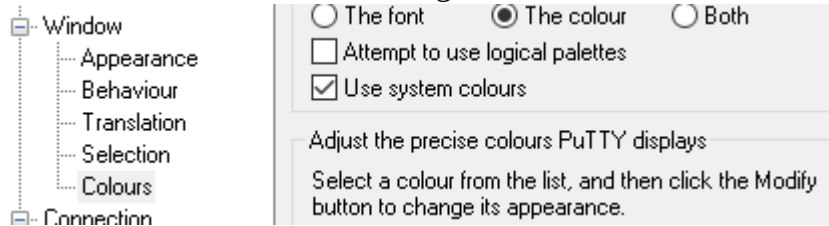


For each configuration settings you've done, just enter a **name** in the *Saved Session* field, and press the **Save** button. In the list below, this **name** should now appear. You can click on it, and press **Load**. – and, after possible other changes, **Open**. Or, without further configuration, double-click to open the session immediately. Of course, you can also **Delete** a session configuration again.

There are other menu points in the Category tree that are relevant. We will not comment on all of them. Some are explained a bit later at the corresponding topics.

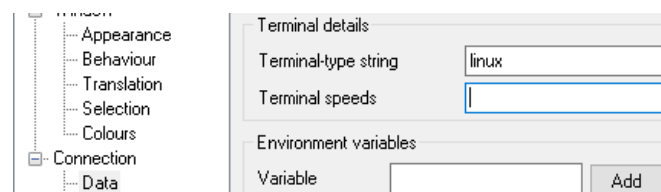
But there are some Default settings that appeared relevant for any convenience such that we mention them here.

1. Colors: White text on black background is is often hard to read.



So, go to *Windows* → *Colors*, and make a hook on *Use system colours*! This creates a white background with black text. You can change this also during the session. But to have it permanent, make this setting and *Save* the corresponding session configuration.

2. LRZ login hosts all have Linux Operating Systems. The native terminal does not play so nicely with them (for instance, *Pos1* and *End* are not recognized). To change this, go to *Connection* → *Data*, and change the fields under *Terminal details* as shown in the following figure.

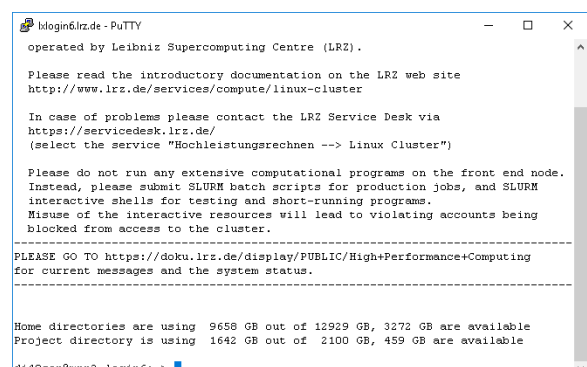


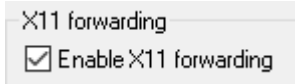
To make it permanent, *Save* again the session configuration.

The terminal window that opens should now look as follows.

X-Forwarding

If you have a running X-server (under Windows, this might be [Xming](#); Except for *MobaXterm*, this requires



usually admin rights³) you can try simple graphics forwarding. For this to work, configure your session under *Connection* → *SSH* → *X11* with a hook . For permanence,

Save the configuration again!

On **success**, you should now be able to open a graphical program! For instance, start the PDF viewer *evince*:

```
> evince
```

or

```
> module load paraview
```

```
> paraview
```

(Firefox might not work, because it is too clever and opens YOUR LOCAL Browser, if present!) When you finished. Close the graphical window, and log out!

SSH Hopping

Via SSH, one can connect also to other (SSH server) hosts, which are e.g. behind a firewall (such as ours, represented by the login nodes). As this must be accomplished via the OpenSSH client on the login nodes, please consult the OpenSSH tutorial under *SSH Hopping and ProxyJump*, where latter is to our knowledge not possible directly using PuTTY.

If you need more automated SSH connection, the PuTTY suite also contains a command-line tool called [plink](#). It is a little bit more advanced, and thus not part of this tutorial.

Copying Data

pscp

pscp works on the command-line of Windows, similar as *plink* does. Because there are very convenient alternatives, we skip the description of this tool here. If you are interested, ask the instructor in a break!

Filezilla

Filezilla is a GUI based SSH/SCP client, available for Linux, Mac (Apple), and Windows. It is also available as portable app, and requires no administrator for installation.

Figure 2 shows the client after opening. In the menu line, you must enter the server name to which you want to connect, the userID, the password, and port 22 (usually).

After successful connection, you can copy files and folders via drag-and-drop. This is surely more convenient than SCP.

3 We show later alternatives!

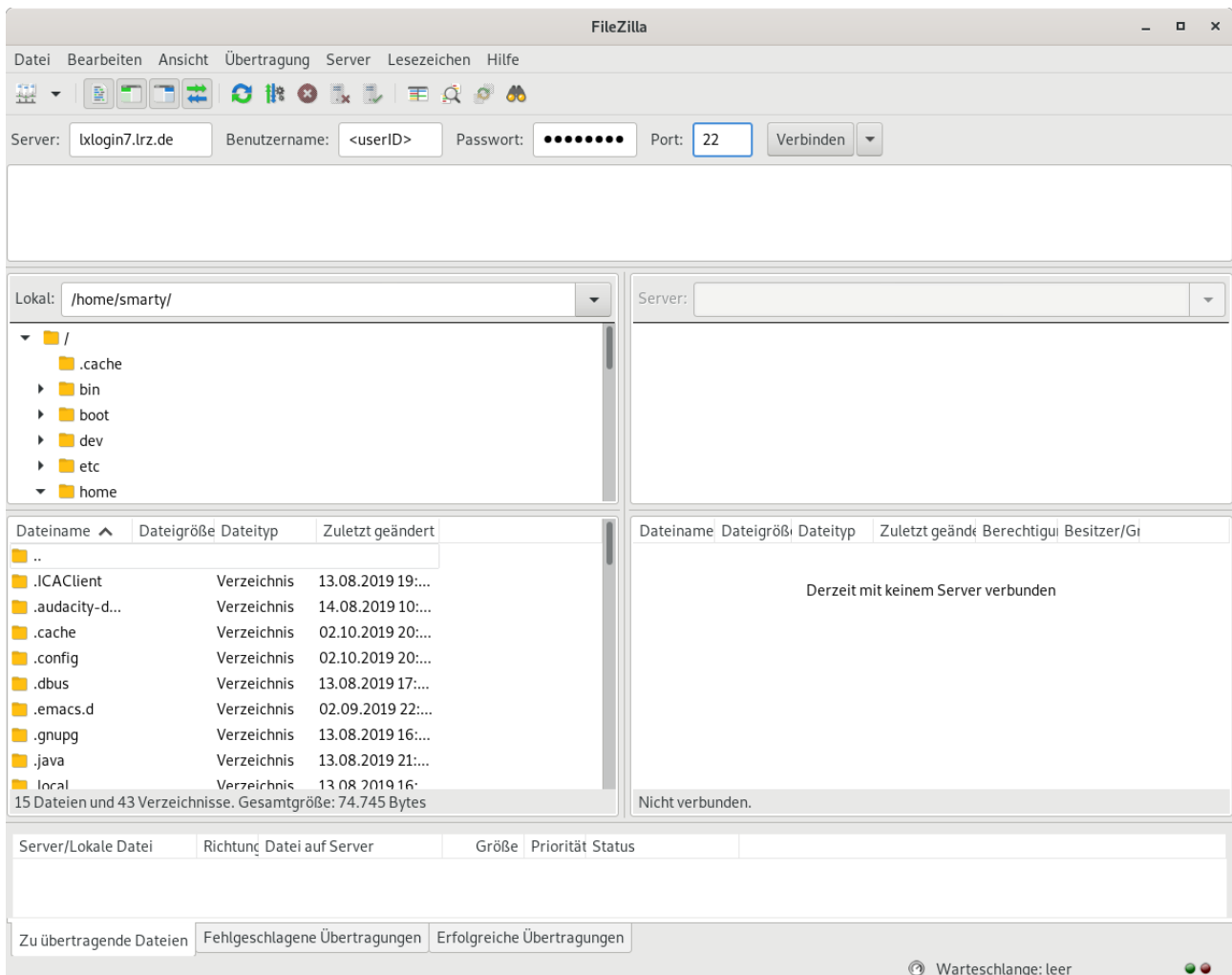


Figure 2: Filezilla GUI as drag-and-drop client

WinSCP

Another GUI based tool for copying data between hosts is [WinSCP](#). Also this tool exists as portable app, which does not require administrative rights for an installation.

Public Key Authentication

Up to now, we authenticated only via our userID and the corresponding password. This method bears several problems – among others that observers might espy (maybe even only accidentally) your credentials while you hack them into your keyboard. Or if you are inattentive, you might accidentally write it to the bare screen – readable for every eye.

Finally, if you are logging in and out very frequently, possibly over several hosts, and maybe also as different users on different hosts with each having a different password, entering credentials always by hand will drive you crazy sooner or later. Latest now, you should wonder whether a security feature should not be more convenient to useful! And the answer is, of course, ‘Yes!’.

Using SSH public-private key pairs is the answer. Their checkout and usage can be automated (looks like password-less login). You don’t need to remember all the passwords (except for the cases, where you really need them!). SSH keys are really long. And so much more secure than your password can possibly ever be! And even if some person acquires your SSH key (specifically the private one), he/she cannot guess your password itself – and he/she has only access to the system

for which this key was generated. Once you have the suspicion that your account was hacked in this way, just disable your key! You have the full control over your account and the keys – you can do hardly anything wrong!

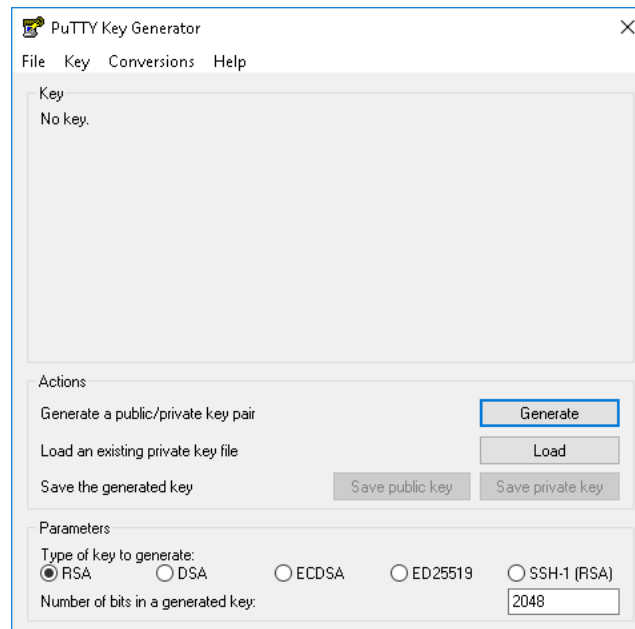
Finally, there are some systems like Cloud VMs, which can only be access via SSH keys. Here, you even have no other choice!

There are two stages. 1) Creation of keys. 2) Distribution of Keys. 3) Using the keys. For the last to be convenient, we introduce the concept (and tool) of an SSH agent.

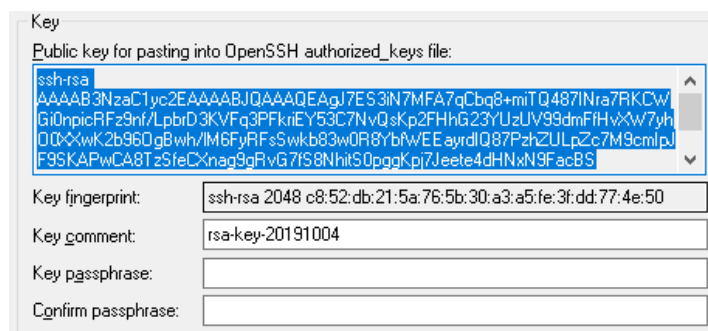
Creation and distribution of keys you usually does only seldom. Of course, you could reuse a created key pair for several independent remote hosts (with possibly different user IDs). However, from a security point of view, you should not do this.

Key Generation

To create a key, we use the tool PuTTYgen. Its usage is nearly children's game.




Make your settings (type and number of bits), and press **Generate**. Next, move your mouse in the field until the green progress bar completes. Now, you should see a generated key.



You need to copy it (mark it and 'Ctrl+C'), open a PuTTY session to the remote SSH server, and insert it into the `~/.ssh/authorized_keys`.

```
> echo "ssh-rsa AAAAB3N..." >> ~/.ssh/authorized_keys
```

Take care to use YOUR key (not that of the tutorial)! Also be careful to use >>! With a single > you overwrite the file, removing everything what was in before.⁴ After finishing, logout again!


Finally, in PuTTYgen, enter key passphrase in the corresponding fields⁵, and press  to save the private key! Keep this file secured from outside access!

The key passphrase is NOT any password you already have! It is a new passphrase you should enter. But in contrast to your SSH password, it does not need to be all too strong. Its only purpose is to protect your private key! So, please, don't leave it empty!⁶ Some basic notion of security must be there in all this convenience! And don't reuse it! (see [Bruce Schneier's article](#) on creating good passwords/passphrases.)

Next to RSA, you can also choose other keys like ECDSA or ED25519. If servers (for security reasons) do not accept certain encryption method anymore (because, they might be compromised by newer computer development), then you must change to a new/different algorithm. This also means that you should keep your SSH suite up-to-date.

For LRZ, please use ECDSA 521 bit, or ED25519!

Key Usage

Once your keys are created and correctly distributed, you can test the access. In order for a key to be active, you must start *Pageant* . It most probably opens as tray icon without any other notice. Right-click on it and choose *Add key* or *View keys* → *Add key*. Select the private key file you saved before! You are now asked for the passphrase. Correctly entered, your key is now active.

For testing, just open a session to the server where you put the public key to (our login nodes have the HOME directories of users in common via a shared files system – so you can login to any of the login nodes).

Troubleshooting and Recovery

If you fear that your key was compromised, or you simply forgot the passphrase, and want to start anew, it does not suffice to delete just the local private key file and from pageant. You must also remove the corresponding public key from the remote `~/.ssh/authorized_keys`!

Exercise: Create keys, and remove them again. Some times should suffice to gain a feeling how it works!

Summary

1. Create keys via *PuTTYgen*, save the passphrase-protected private key in a local file, and copy the public key into the remote `~/.ssh/authorized_keys`!
2. For regular usage, just start *Pageant*, and add the key! Ready to start SSH! Some clients like Filezilla and WinSCP might be able also to use these keys.

⁴ This is bash output redirection! You should learn bash anyway, asap!

⁵ You are asked twice for this key to ensure you typed it correctly.

⁶ Keys used to access the LRZ systems from outside MUST be passphrase protected (that's Policy)!

Port Forwarding / SSH Tunneling

The last part deals with some more advanced, and really cool stuff. SSH connections admit it to tunnel through them other protocols like HTTP or VNC (if this sounds familiar to you). Most beginners with SSH don't like to work through a terminal or in a shell. The following may help here to mitigate the first pains.

For the following to work, you must remember what ports are (for what they are useful), and that only one server on a system can acquire a port – and block it from the usage by others. And all ports below 1024 are privileged ports – users can't use them! All other ports up to 65535 can be used.

Preparation

You can use any (or all) of the following examples. If a port is blocked, use another one! Login to one `lxlogin*` node via PuTTY and start one of the following servers.

1) *Webbrowser: (requires a local Browser)*

```
> module load python/3.8.8-base      # some python 3
> cat > index.html << EOF
<html>
  <head>
    <title>Hello LRZ!</title>
  </head>
  <body>
    <h1>Welcome to my Webpage!</h1>
  </body>
</html>
EOF
> python -m http.server 11111
Ctrl+C          # kill the server when finished the exercise
```

2) *Jupyter Notebook: (requires a local Browser)*⁷

```
> module load python                # or install conda
> pip install jupyter
> jupyter-notebook --port=11111 --no-browser --ip=localhost
The Jupyter Notebook is running at: http://localhost:11111/?
token=3eef5acb44b47f385be6dcdd341b319d4d13fdc3879c6c9f
Ctrl+C          # kill the server when finished the exercise
```

3) *VNC Server: (requires a local VNC Viewer)*⁸

```
> vncserver -list                   # check empty ports
> vncpasswd                          # choose a simple password!
> vncserver :5                       # for port 5900+5=5905
> vncserver -kill :5                # kill server when exercise finished
```

Next, once one of your servers is running, we try to connect to it. Keep the server running during this time. Remember on which **host** and on which **port** your server runs! We need this information!

⁷ Check <https://doku.lrz.de/display/PUBLIC/FAQ:+Jupyter+on+HPC+Systems>

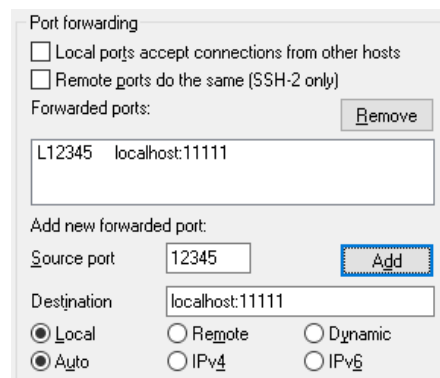
⁸ **Please check for more infos:** <https://doku.lrz.de/display/PUBLIC/VNC+Server+on+Login-Nodes>

Tunnel and Connection to the same Server

The first scenario is the simpler one. We connect to the SSH Server, where also the HTTP/VNC Server is running. This is maybe the more often case, as `lxlogin*` servers are the login servers, where you may also run such smaller applications (for the larger ones, the Slurm Cluster is dedicated!).

As you have a PuTTY session already open, right-click on the terminal's title bar (top of the window). A drop-down menu should appear. Select *Change Settings...*! A PuTTY (re)configuration window opens. As you can see, there are not all entries in the category tree available anymore – it is a running session. But colors and font-sizes still can be changed.

To create a tunnel go to *Connection* → *SSH* → *Tunnels*, and enter the respective data into the fields.



It is important to press **Add** such that the tunnel appears correctly in the list of tunnels attached to this SSH connection.

Such tunnels can in principle already setup before the PuTTY session is opened. And thus, they can be also saved in a configuration. You but usually have to clue whether the ports are free before you test.

Here, 12345 is a arbitrary local port.⁹ As remote port, we specified here [11111](#). Please select the port, you chose before for your server (e.g. [5905](#) for the VNC server above)!

Depending on the example chosen do the following.

- 1) Open a Browser with `http://localhost:12345`
- 2) Open a Browser with the webaddress shown on the servers output, but port 11111 replaced by the local port 12345. `http://localhost:11111/?token=3eef5acb44b47f385be6dcdd341b319d4d13fdc3879c6c9f`
Specifically the token (you have to take your own token, of course) is important to get access to your server!
- 3) Open the VNC viewer with `localhost:12345`. You must enter the VNC password.

The usage of browsers and VNC is not part of this course. But I'm sure you already recognize what you can use it for.

Please, when finished, stop all servers on the login nodes again!

⁹ Must be also unique. If you have several tunnels open, all their local ports must be different! It is sometimes hard to keep an overview. In the worst case, close all tunnels and SSH connections, and start from the beginning!

A final remark. Instead of `localhost` in the tunnel specification, you can also specify another host name, e.g. from a server behind the firewall of the login nodes. The remote visualization is such an example. In this way, you can also tunnel any protocol through a firewall.

If you want to test this, please ask the instructor!

Appendix

Useful/Entertaining Links

- https://www.schneier.com/blog/archives/2014/03/choosing_secure_1.html
- <http://cryptocouple.com/>
- <http://web.mit.edu/jemorris/humor/alice-and-bob>