Updated August 29, 2017

HOW TO USE PUTTY ON WINDOWS

This page is about the PuTTY SSH client on Windows. For information about PuTTY on Mac, see the PuTTY Mac page. For PuTTY on Linux, see the PuTTY Linux page.

This page explains how to use the PuTTY terminal window on Windows. How to configure PuTTY, how to create and save profiles, and what configuration options to change. Advanced topics, such as configuring public key authentication, are also addressed.

Contents

GETTING AND INSTALLING

You can download a copy of the software for the Windows platform from the download page. Detailed installation instructions are provided on the installation instructions page.

RUNNING PUTTY AND CONNECTING TO A SERVER

If you selected to create a desktop icon during installation, you can start the software simply by (double-)clicking on the icon. Otherwise, open the software from the Windows **Start** menu.

When the software starts, a window titled **PuTTY Configuration** should open. This window has a configuration pane on the left, a **Host Name (or IP address)** field and other options in the middle, and a pane for saving session profiles in the lower right area.

For simple use, all you need to do is to enter the domain name or IP address of the host you want to connect to in the **Host Name** field and click **Open** (or press Enter). A domain name looks like `students.example.edu`. An IP address looks something like `78.99.129.32`.

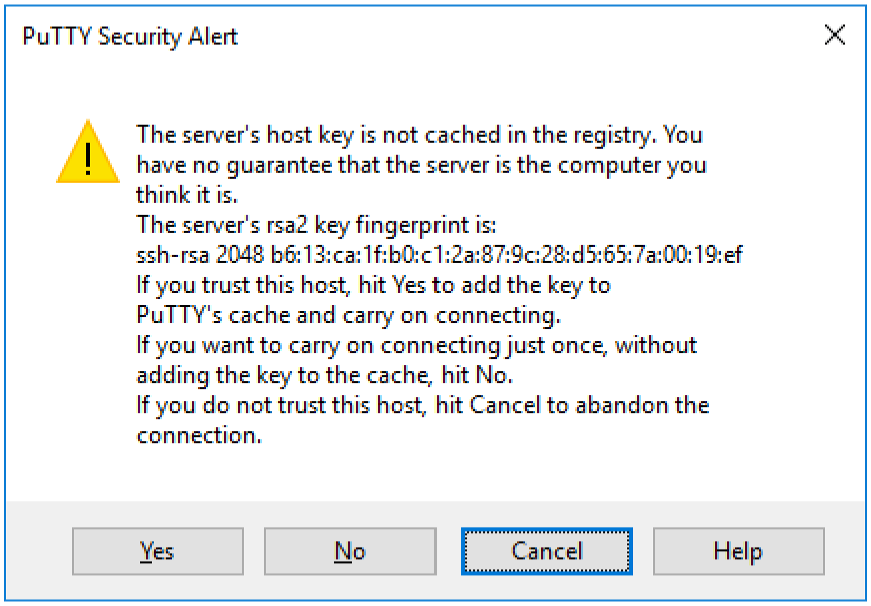
WHAT IF YOU DON'T HAVE A SERVER

If you don't have a server to connect to, you can try Tectia SSH on Windows or OpenSSH on Linux.

SECURITY ALERT DIALOG BOX

When you connect to a server for the first time, you are likely to see a `PuTTY Security Alert` dialog about the server's host key not being cached in the registry. This is normal when you are connecting to a server for the first time. If you ever get this with a server, it could mean that someone is trying to attack your connection and steal your password using a man-in-the-middle attack.

But as said, the first time you connect, this is normal, and you should just click **Yes**. If you want to be fancy, you can check the displayed key fingerprint and make sure it is the same that is used by the server. In real life, almost nobody does that. It is more secure to use a proper SSH key management solution anyway.

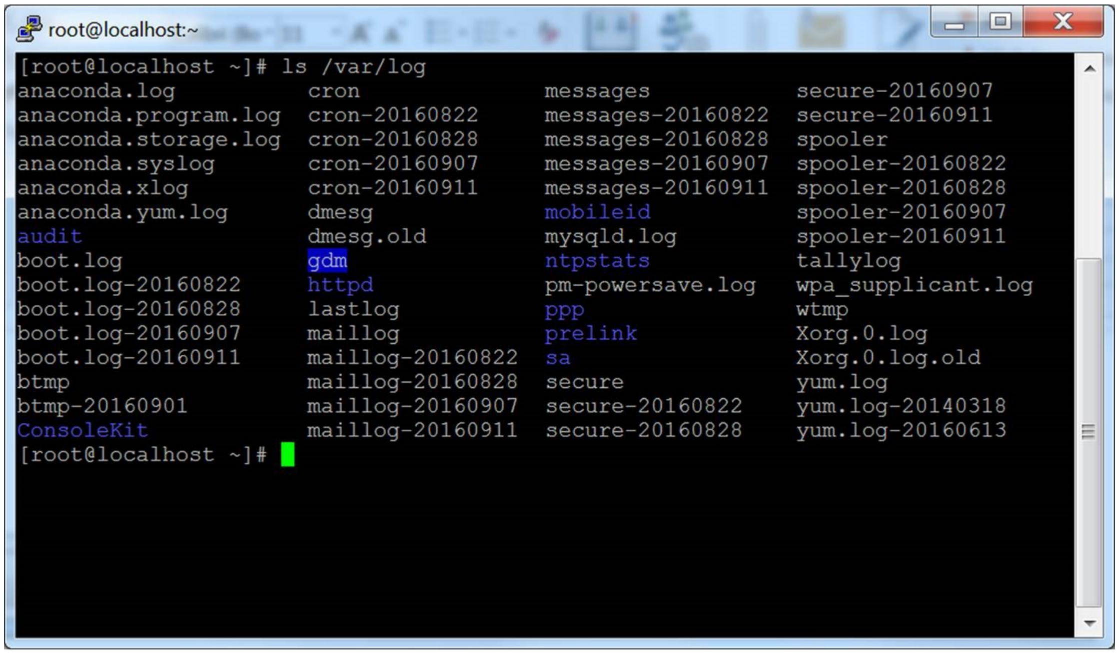

PuTTY Security Alert

The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048 b6:13:ca:1f:b0:c1:2a:87:9c:28:d5:65:7a:00:19:ef
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

TERMINAL WINDOW AND LOGIN CREDENTIALS

After the security alert, you should get a terminal window. By default, this is a black, very bland window. It should first ask for your user name and then password. After these, you should get a command line on the server.

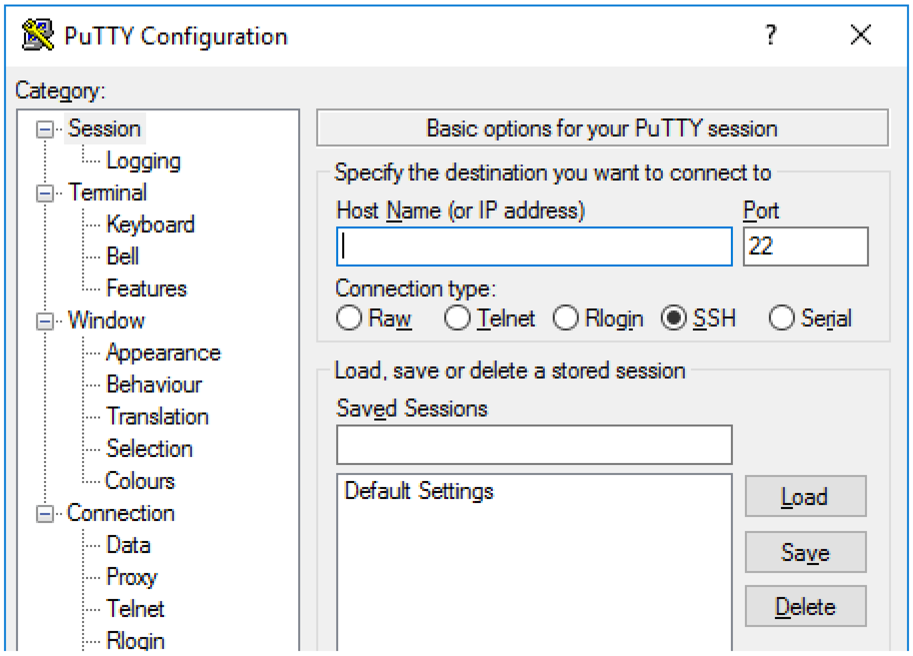
You can then type into the terminal Window. You are now connected to the server, and anything you type in the Window is sent to the server. Server's responses are displayed in the Window. You can run any text-based applications on the server using displayed in the window. You can run any text-based applications on the server using the window. The session terminates when you exit the command-line shell on the server (typically by typing `exit`) to the command line or pressing `Control-D`. Alternatively, you can forcibly terminate the session by closing the terminal window.



```
[root@localhost ~]# ls /var/log
anaconda.log          cron            messages           secure-20160907
anaconda.program.log  cron-20160822   messages-20160822  secure-20160911
anaconda.storage.log  cron-20160828   messages-20160828  spooler
anaconda.syslog       cron-20160907   messages-20160907  spooler-20160822
anaconda.xlog         cron-20160911   messages-20160911  spooler-20160828
anaconda.yum.log      dmesg           mobileid           spooler-20160907
audit                 dmesg.old       mysqld.log         spooler-20160911
boot.log              gdm             ntpstats           tallylog
boot.log-20160822     httpd           pm-powersave.log   wpa_supplicant.log
boot.log-20160828     lastlog         ppp                wtmp
boot.log-20160907     maillog         prelink            Xorg.0.log
boot.log-20160911     maillog-20160822 sa                Xorg.0.log.old
btmp                  maillog-20160828 secure            yum.log
btmp-20160901         maillog-20160907 secure-20160822   yum.log-20140318
ConsoleKit            maillog-20160911 secure-20160828   yum.log-20160613
[root@localhost ~]#
```

CONFIGURATION OPTIONS AND SAVED PROFILES

The initial configuration window contains a lot of options. Most of them are not needed in normal use.



PuTTY Configuration

Category:

- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address):

Port: 22

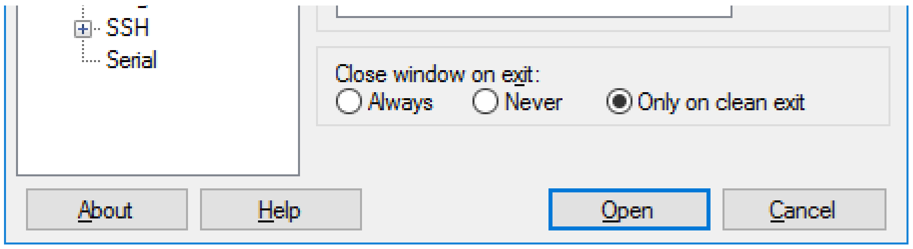
Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions

Default Settings

Load Save Delete



Close window on exit: Always Never [✓ Only on clean exit]

PORT

The **port** field specifies the TCP/IP port to connect. For SSH, this is the port on which the SSH server runs. Normally it can be left to 22. If for some reason you need to connect to a different port number, just change the value. Usually only developers would change this to a different value, but some enterprises are known to run SSH servers in non-standard ports or to run multiple SSH servers on the same server at different ports.

CONNECTION TYPE

The **Connection type** selection almost never needs to be touched. Just leave it as **SSH**. SSH is a secure, encrypted communications protocol designed to ensure your password and data are maximally protected.

Raw connections might be used for developers to connect a TCP/IP socket for testing (e.g., when developing a network application that listens on a TCP/IP port).

Telnet is an old legacy protocol that is almost never used, unless you manage equipment that is more than 10 years old. Telnet is not secure. Passwords are sent in the clear on the network. Attackers can easily eavesdrop on plaintext communications and steal user names and passwords. **Rlogin** is another legacy protocol with similar woes.

Serial refers to a serial port, another legacy communications mechanism for connecting computers to peripheral devices. Most PCs these days no longer have serial ports, but they are still sometimes used for controlling physical equipment, instrumentation, machinery, or communications devices. Another use for serial ports is debugging operating systems or embedded software.

LOAD, SAVE, OR DELETE A STORED SESSION

This section allows you to save your settings as named profiles. Just write the name of your new profile in the **Saved Sessions** box and click **Save** to create a new profile. The host name and your other settings are saved in the profile.

Saved profiles appear in the larger box below it. Initially it will contain just **Default Settings**. Profiles you save will be included there. Select a profile and click **Load** to use a previously saved profile. Select a profile and click **Delete** to delete a profile that is no longer needed.

CLOSE WINDOW ON EXIT

Finally, the **Close window on exit** setting specifies whether the terminal window should be automatically closed when the connection is terminated. There is rarely any need to change it from the default value of **Only on clean exit**.

LEFT PANE CONFIGURATION OPTIONS

More options can be found in the left pane titled **Category**. Select a category from the tree, and the right pane will change to show configuration options for that category. The initally shown options belong to the **Session** category.

Only the more relevant options are described here. There are lots of options, and most of them would never be used.

TERMINAL OPTIONS

The options in this category influence terminal emulation and keyboard mappings. They are largely self-explanatory, and will not be covered here. Very few people need to touch these. Some people may change how the **bell** character is handled; people using exotic operating systems might change what is sent by the **backspace** or **delete** character.

WINDOW OPTIONS

The window options influence the appearance and behavior of the terminal window. It can also specify how characters are translated on output and to select fonts and colors for the window.

CONNECTION OPTIONS

Of the connection options, the **Data** options can be useful. The **Auto-login user name** specifies the user to log in as, so that the name will not have to be entered every time. The **Proxy** options are rarely useful for home users, but may be needed in enterprises that do not allow outgoing Internet connections without using a SOCKS proxy or other similar mechanisms. Don't worry if you don't know what a SOCKS proxy is; just stay out of that section.

The **Telnet**, **Rlogin**, and **Serial** categories only contain options for those protocols, and very few people would ever use them.

The **SSH** options, however, are important and useful for some people. The ordinary user or student need not worry about them. But if you want to use public key authentication, then they are needed. Note that **you need to open the SSH options subtree** by clicking on the small [+] symbol. Otherwise you won't see all the options.

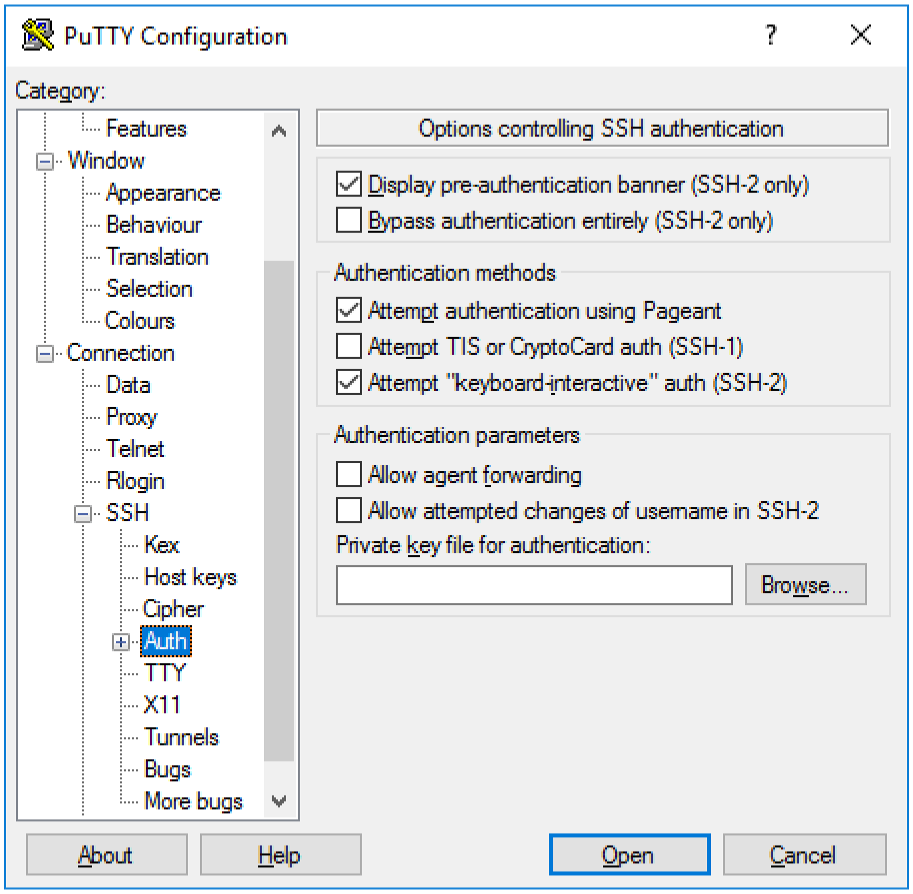
KEY EXCHANGE, HOST KEYS, AND CIPHER OPTIONS

You almost never want to touch the Kex (key exchange), Host Keys, or Cipher options. They all have reasonable default values, and most people don't know enough about cryptography to select any better values. Thus just skip these options, unless you know what you are doing.

AUTHENTICATION OPTIONS - PUBLIC KEY AUTHENTICATION

The **Auth** subtree contains some options that may be useful. When **Auth** is clicked, it shows a pane titled **Options controlling SSH authentication**. To enable public key authentication, you just generate an SSH key and then click the **Browse** button in the **Authentication parameters** box in the middle right area of this configuration pane. For more information, see also configuring public key authentication for PuTTY. Advanced users may also want to check the **Allow agent forwarding** checkbox to use key-based single sign-on.

Most users have no need to generate SSH keys and need not know what public key authentication is. System administrators, however, should learn it and should also familiarize themselves with SSH key management and ensure their organization implements proper provisioning and termination processes and audits for SSH keys.



Options controlling SSH authentication

- [✓] Display pre-authentication banner (SSH-2 only)
- [ ] Bypass authentication entirely (SSH-2 only)

Authentication methods

- [✓] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [✓] Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

- [ ] Allow agent forwarding
- [ ] Allow attempted changes of username in SSH-2

Private key file for authentication:

ACTIVE DIRECTORY AUTHENTICATION (GSSAPI / KERBEROS)

One of the interesting features of PuTTY is support for Active Directory single sign-on. Technically it uses the Kerberos protocol via a programming interface called GSSAPI. In the SSH protocol, the mechanism is called GSSAPI authentication. Enterprise users using Kerberos authentication (e.g., via the Centrify or Quest Authentication Services aka Vintela) may want to take advantage of the single-sign-on capability. Other users don't need to care. The settings for GSSAPI authentication can be found under the **SSH** / **Auth** section. Note that you must again expand the **Auth** section by clicking on the [+] symbol to see the GSSAPI options.

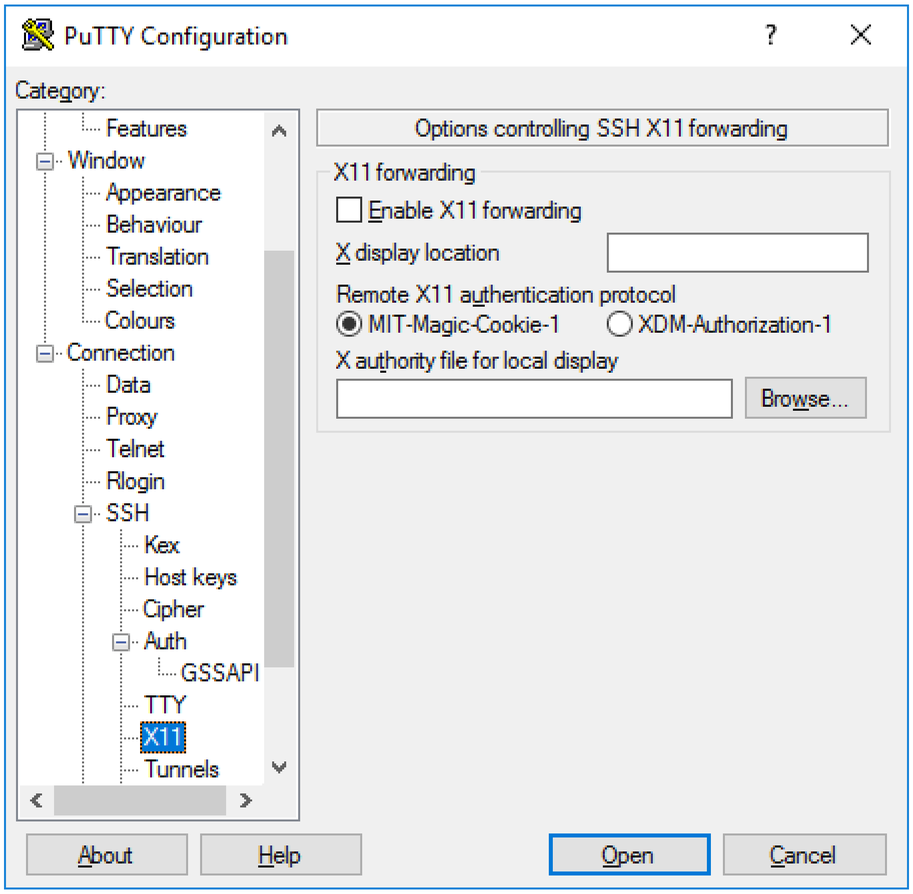
X11 FORWARDING OPTIONS

X11 is a protocol and system for running graphical applications on Unix and Linux. It supports running graphical applications remotely over a network out-of-the-box.

PuTTY does not implement an X11 server (the display side), but it can work with

PuTTY does not implement an X11 server (the display side), but it can work with some other product that implements X server functionality on Windows. A popular free alternative is XMing.

To use an X11 server, you need to check the **Enable X11 forwarding** box and enter **localhost:0.0** in the **X display location** box. The other settings need not be touched. Detailed instructions can be found, e.g., here.


PuTTY Configuration

Category:

- Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
    - Kex
    - Host keys
    - Cipher
    - Auth
      - GSSAPI
    - TTY
    - X11
    - Tunnels

Options controlling SSH X11 forwarding

X11 forwarding

☐ Enable X11 forwarding

X display location

Remote X11 authentication protocol

[✓ MIT-Magic-Cookie-1] XDM-Authorization-1

X authority file for local display

TUNNELING OPTIONS

The final category of configuration options we'll discuss is **Tunnels**. They are used for configuring SSH tunneling, also called SSH port forwarding. This panel can be used for defining forwardings for the connection. Forwardings are saved in the profile.

To add a local forwarding (i.e., TCP/IP port on local machine forwarded to a port on the remote machine or to a machine reachable from the remote machine), write the source port in the **Source port** field, the destination host and port (e.g., `www.dest.com:80`) in the **Destination** field, and select **Local**. The click **Add**.

To add a remote forwarding (i.e., a TCP/IP port on the remote machine forwarded to a port on the local machine or to a machine reachable from the local machine), a port on the local machine or to a machine reachable from the local machine), specify **Source port** on the destination machine and **Destination** that is reachable from the local machine (your desktop).

Normally you need not check **Local ports accept connections from other hosts** or the same for remote ports. However, if the connection to the forwarded port is from a over a network instead of from `localhost`, then you need to check these. There is a small security risk, but usually it is not a problem in the cases where SSH tunneling is used. However you should understand that anyone who can connect to the respective computer can then connect to the forwarded port. In some cases port forwarding can be used to traverse firewalls. We suggest you read our article on the risks of SSH port forwarding.