**Enable X11 forwarding**

VNC is faster than X. you should use VNC instead of X

While doing ssh use the option -X to enable X11 forwarding.

$ ssh username@hostname -X

Enable trusted X11 forwarding, by using the -Y option,

$ ssh username@hostname -Y

The OpenSSH client supports two levels of X11 forwarding, differentiated by desired security level. Configure both in ssh\_config.

Basic X11 forwarding supports only a less-insecure subset of the X11 protocol. This level of X11 forwarding is fairly safe. Intruders cannot, say, take over your desktop or snoop your keystrokes with this level of X11 forwarding.

ForwardX11 yes

Always try this basic X11 forwarding first.

Many programs use X functions other than the secure subset. When forwarded over SSH, these programs show an error and crash. You can choose to allow the full set of X functions with the option ForwardX11Trusted.

ForwardX11Trusted yes

Trusting X11 permits all X functions. An intruder on the SSH server can capture everything on your local screen and your every keystroke. Be really, really sure you trust every single remote server you might ever log into before permitting this level of trust globally. And once you're absolutely certain – don't do it

**Encrypting X Connections with SSH**  
Suppose your local network contains two machines. The computer called 192.168.100.101 is a powerful machine that hosts important programs like word processors and data analysis utilities.  
The computer called 192.168.100.102 is a much less powerful system, but it has an adequate monitor  
and keyboard. Therefore, you want to sit at 192.168.100.102 and run programs that are located on  
192.168.100.101. Both systems run Linux. To accomplish this task, follow these steps:  
**1.** Log into 192.168.100.102, and if it’s not already running X, start it.  
**2.** Open a terminal (such as an xterm) on 192.168.100.102.  
**3.** Log into 192.168.100.101 from 192.168.100.102. You might use Telnet or Secure Shell (SSH), for instance.  
The result should be the ability to type commands in a shell on 192.168.100.101.

ssh –X root@192.168.100.101

**4.** Type whatever you need to type to run programs at the 192.168.100.101 command prompt. For  
instance, you could type **loffice (or firefox)** to launch LibreOffice. You should see the programs  
open on 192.168.100.102’s display, but they’re running on 192.168.100.101—their computations use 192.168.100.101’s  
CPU, they can read files accessible on 192.168.100.101, and so on.

Note:  
The SSH protocol is a useful remote-access tool. Although it’s often considered a textmode protocol, SSH also has the ability to *tunnel* network connections—that is, to carry  
another protocol through its own encrypted connection. This feature is most useful for  
handling remote X access. You can perform the steps described in “Using Remote X  
Clients” but omit steps 3 and 5 and the xhost command in step 7. This greatly simplifies  
the login process and adds the benefits of SSH’s encryption, which X doesn’t provide. On  
the other hand, SSH’s encryption is likely to slow down X access, although if you enable  
SSH’s compression features, this problem may be reduced in severity. Overall, tunneling  
X through SSH is the preferred method of remote X access, particularly when any network in the process isn’t totally secure.  
SSH tunneling does require that certain options be set. In particular, you must either use  
the -X or -Y option to the ssh client program or set the ForwardX11 or ForwardX11Trusted  
option to yes in /etc/ssh\_config on the client system. You must also set the X11Forwarding option to yes in the /etc/sshd\_config file on the SSH server system. These options  
enable SSH’s X forwarding feature; without these options, SSH’s X forwarding won’t work.