Scotti-BYTE Enterprise Consulting Services

How to Leverage X-11 Across Windows and Macintosh

In a recent tutorial, I discussed Windows Subsystem for Linux 2 (WSL2) which is basically a Linux kernel and shell that can operate inside of Windows 10. In the end of that article, I mentioned VcXsrv (https://sourceforge.net/projects/vcxsrv/) which is an open source Windows X-server.

In the Linux world, the X Window System (also known as X11, or just X) is a software package and network protocol that lets you interact locally, using your personal computer's display, mouse, and keyboard, with the graphical user interface (GUI) of an application running on a remote networked computer. VcXsrv adds an otherwise non-existent X-Windows server to a Windows computer.

In fact, on Linux anytime that you run a program that opens a graphically displayed window, it is using the X-Windows protocol. X-Windows has the concept of a client and a server. The server is the place where the data is displayed and the client is the computer where the program is running. So, on a Linux machine if you bring up a file manager screen, the file manager program is running on the client which just happens to be the local machine and it displays on the local machine because it is acting as the server.

In linux, this can easily be changed. You can open a terminal and run the following command:

```
export DISPLAY=172.16.1.186:0.0
```

This would change the X-Windows display for the current terminal to where any commands that follow would try to display their output on a X-Windows server on a machine at address 172.16.1.188. The "0:0" at the end means server "0" and screen "0". It is possible, although not common to run more than one X-Windows server on a computer. It might be much more common to issue a command like:

```
export DISPLAY=172.16.1.186:0:1
```

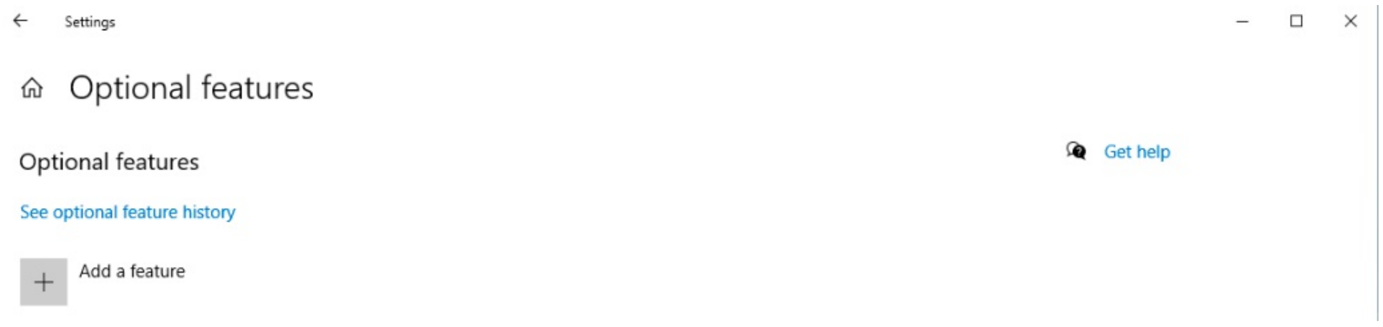
This command would display graphical programs launched subsequently to the computer at 172.16.1.186 but on the right hand screen if the system had two monitors. Screen 0 would be the left hand monitor and Screen 1 would be the right hand monitor.

As you can see, this would be very powerful.

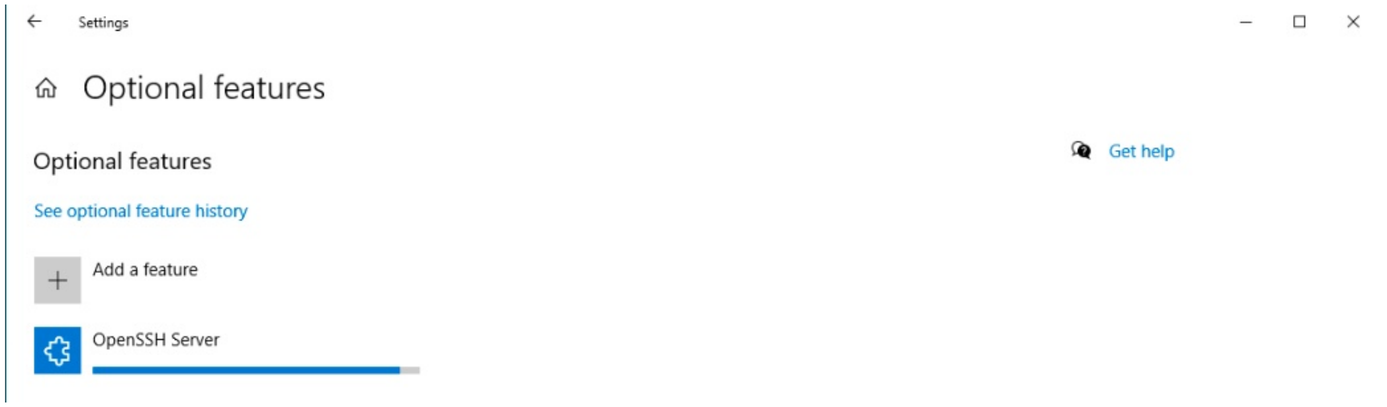
In the majority of my tutorials I make frequent use of the secure shell "ssh" command which allows you to sign on to another computer or submit a single command on another computer. The ssh command requires an "ssh" client to submit commands and an ssh server to receive and execute commands.   To install the "ssh server" in Ubuntu:

```
sudo apt install openssh-server
```

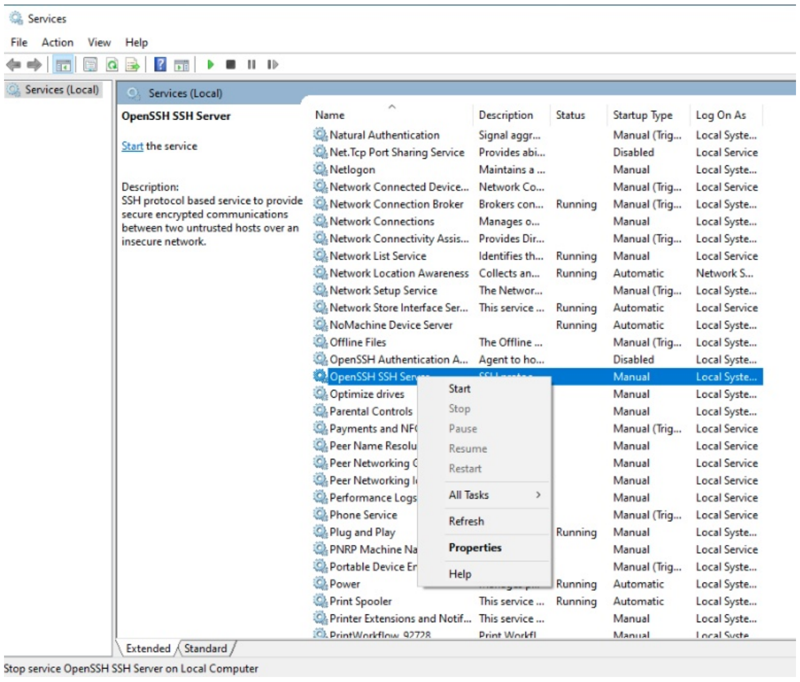
In Microsoft Windows 10, click on settings from the start menu, apps, and then click on optional features.



Click on Add a feature. Scroll down and add "OpenSSH Server" and click on Install.



Now, click on the start menu and type "Services". Scroll down to "Open SSH Server" and Start it.



OpenSSH SSH Server

Start the service

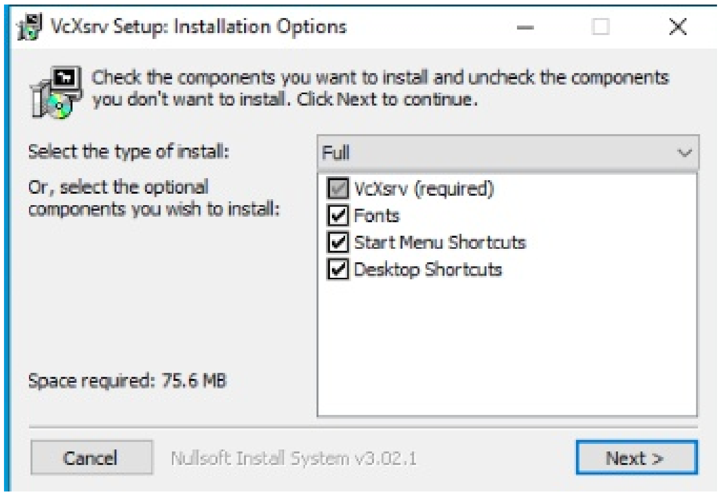
Description:
SSH protocol based service to provide secure encrypted communications between two untrusted hosts over an insecure network.

Name	Description	Status	Startup Type	Log On As
Natural Authentication	Signal aggr...		Manual (Trig...	Local Syste...
Net.Tcp Port Sharing Service	Provides abi...		Disabled	Local Service
Netlogon	Maintains a ...		Manual	Local Syste...
Network Connected Device...	Network Co...		Manual (Trig...	Local Service
Network Connection Broker	Brokers con...	Running	Manual (Trig...	Local Syste...
Network Connections	Manages o...		Manual	Local Syste...
Network Connectivity Assis...	Provides Dir...		Manual (Trig...	Local Syste...
Network List Service	Identifies th...	Running	Manual	Local Service
Network Location Awareness	Collects an...	Running	Automatic	Network S...
Network Setup Service	The Networ...		Manual (Trig...	Local Syste...
Network Store Interface Ser...	This service ...	Running	Automatic	Local Service
NoMachine Device Server		Running	Automatic	Local Syste...
Offline Files	The Offline ...		Manual (Trig...	Local Syste...
OpenSSH Authentication A...	Agent to ho...		Disabled	Local Syste...
OpenSSH SSH Ser			Manual	Local Syste...
Optimize drives			Manual	Local Syste...
Parental Controls			Manual	Local Syste...
Payments and NF			Manual (Trig...	Local Service
Peer Name Resolu			Manual	Local Service
Peer Networking			Manual	Local Service
Peer Networking I			Manual	Local Service
Performance Logs			Manual	Local Service
Phone Service			Manual (Trig...	Local Service
Plug and Play		Running	Manual	Local Syste...
PNRP Machine Na			Manual	Local Service
Portable Device Er			Manual (Trig...	Local Syste...
Power		Running	Automatic	Local Syste...
Print Spooler	This service ...	Running	Automatic	Local Syste...
Printer Extensions and Notif...	This service ...		Manual	Local Syste...
PrintWorkflow_92728	Print Workfl		Manual	Local Syste

Stop service OpenSSH SSH Server on Local Computer

While you are in Services, go to properties on the "OpenSSH Server" and set it to automatic start so that it starts on the next reboot.

Go to (https://sourceforge.net/projects/vcxsrv/) in the web browser on your Windows machine and download and install VcXsrv.


VcXsrv Setup: Installation Options

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select the type of install: Full

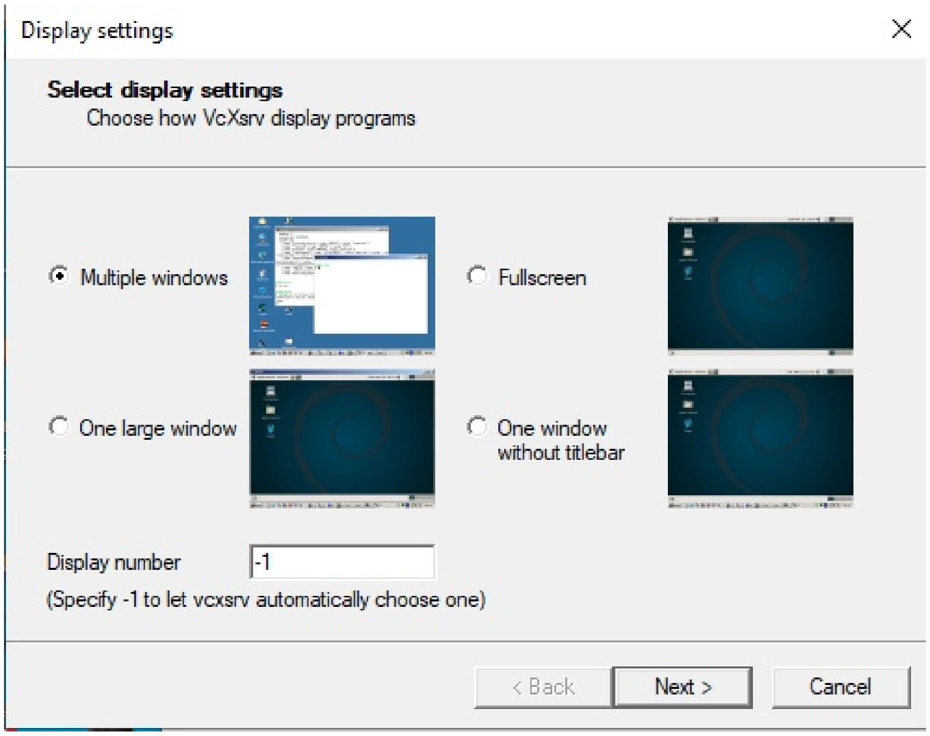
Or, select the optional components you wish to install:

- [✓] VcXsrv (required)
- [✓] Fonts
- [✓] Start Menu Shortcuts
- [✓] Desktop Shortcuts

Space required: 75.6 MB

Nullsoft Install System v3.02.1

After the install completes, you will have a desktop icon named "Xlaunch". Run it:


Display settings

Select display settings

Choose how VcXsrv display programs

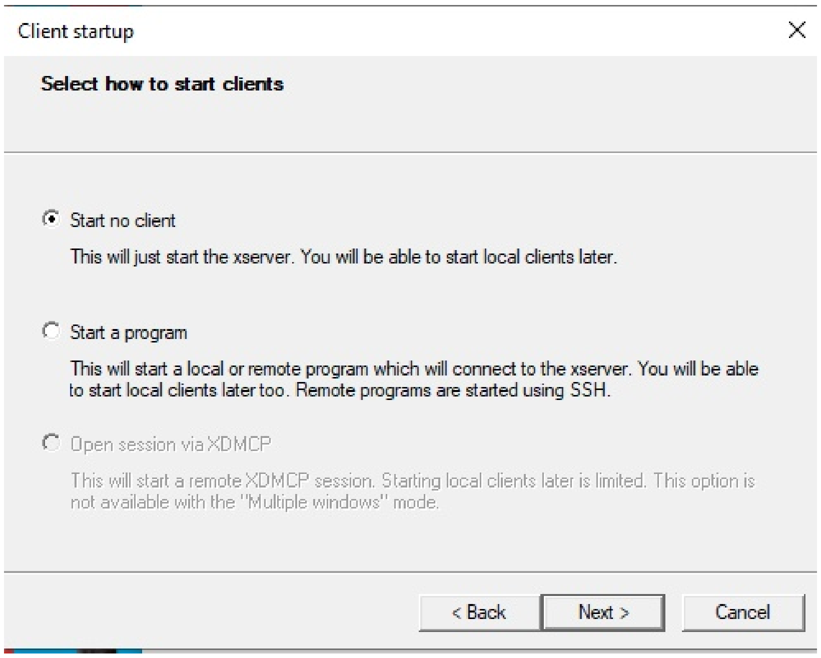
[✓ Multiple windows] Fullscreen

One large window | One window without titlebar

Display number: -1

(Specify -1 to let vcxsrv automatically choose one)

Keep the multiple Windows option and click next.


Client startup

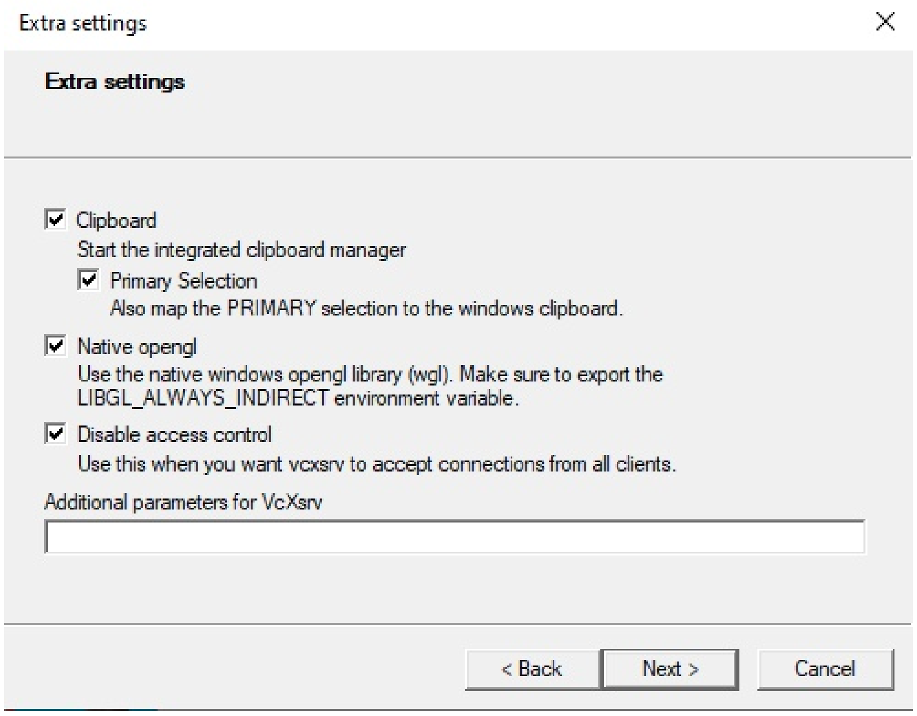
Select how to start clients

(● Start no client) This will just start the xserver. You will be able to start local clients later.

○ Start a program
This will start a local or remote program which will connect to the xserver. You will be able to start local clients later too. Remote programs are started using SSH.

○ Open session via XDMCP
This will start a remote XDMCP session. Starting local clients later is limited. This option is not available with the "Multiple windows" mode.

Choose Start no Client.


Extra settings

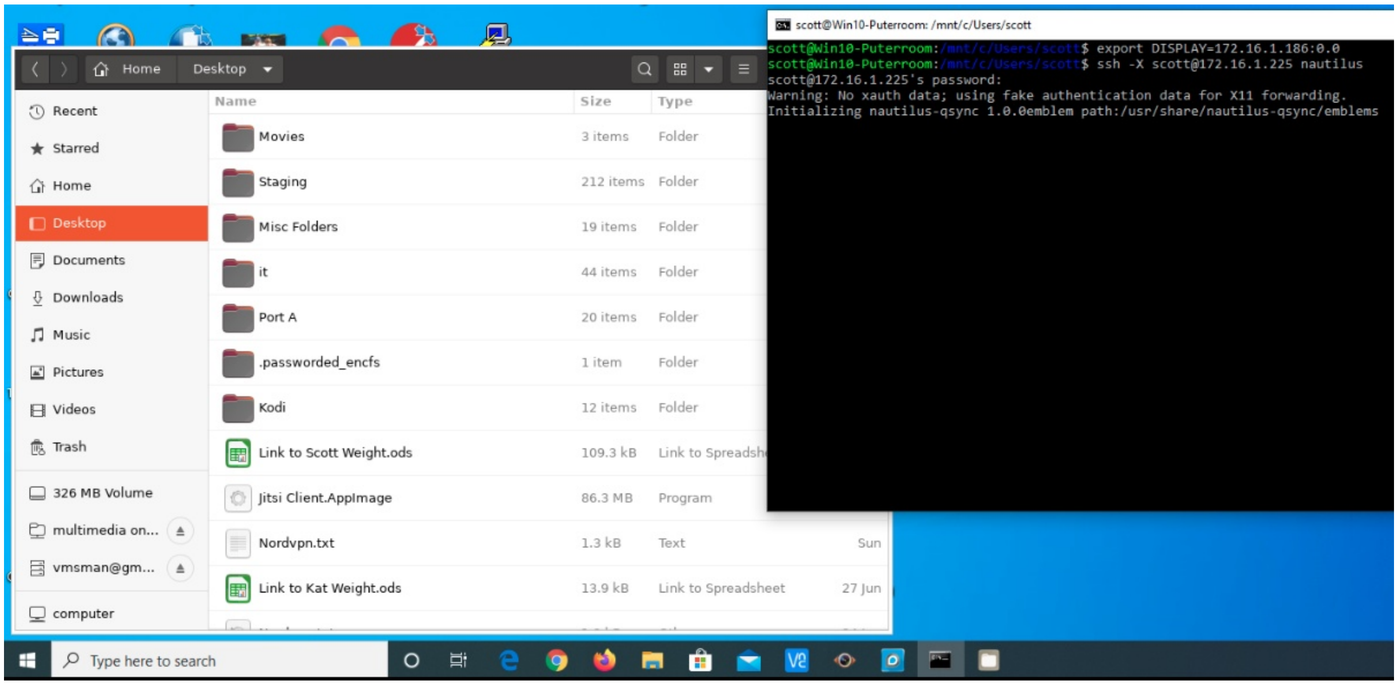
Extra settings

✓ Clipboard
Start the integrated clipboard manager
✓ Primary Selection
Also map the PRIMARY selection to the windows clipboard.
✓ Native opengl
Use the native windows opengl library (wgl). Make sure to export the LIBGL_ALWAYS_INDIRECT environment variable.
✓ Disable access control
Use this when you want vcxsrv to accept connections from all clients.

Additional parameters for VcXsrv

Make sure that Disable access control is checked. These last three screens will come up whenever you start "Xlaunch" in order to launch your X-Server. Note: On initial installation, you might have to tell Windows Firewall to allow this program to communicate on the network.

Since Windows now has an ssh client, an ssh server and an X-Windows server (xlaunch), you have the capability to run programs on another system and display them back to your Windows 10 screen.



Name	Size	Type	
Movies	3 items	Folder	
Staging	212 items	Folder	
Misc Folders	19 items	Folder	
it	44 items	Folder	
Port A	20 items	Folder	
.passworded_encfs	1 item	Folder	
Kodi	12 items	Folder	
Link to Scott Weight.ods	109.3 kB	Link to Spreadsh	
Jitsi Client.AppImage	86.3 MB	Program	
Nordvpn.txt	1.3 kB	Text	Sun
Link to Kat Weight.ods	13.9 kB	Link to Spreadsheet	27 Jun

```
scott@Win10-Puterroom: /mnt/c/Users/scott
scott@Win10-Puterroom:/mnt/c/Users/scott$ export DISPLAY=172.16.1.186:0.0
scott@Win10-Puterroom:/mnt/c/Users/scott$ ssh -X scott@172.16.1.225 nautilus
scott@172.16.1.225's password:
Warning: No xauth data; using fake authentication data for X11 forwarding.
Initializing nautilus-qsync 1.0.0emblem path:/usr/share/nautilus-qsync/emblems
```

In the Windows WSL terminal above, I issued an:

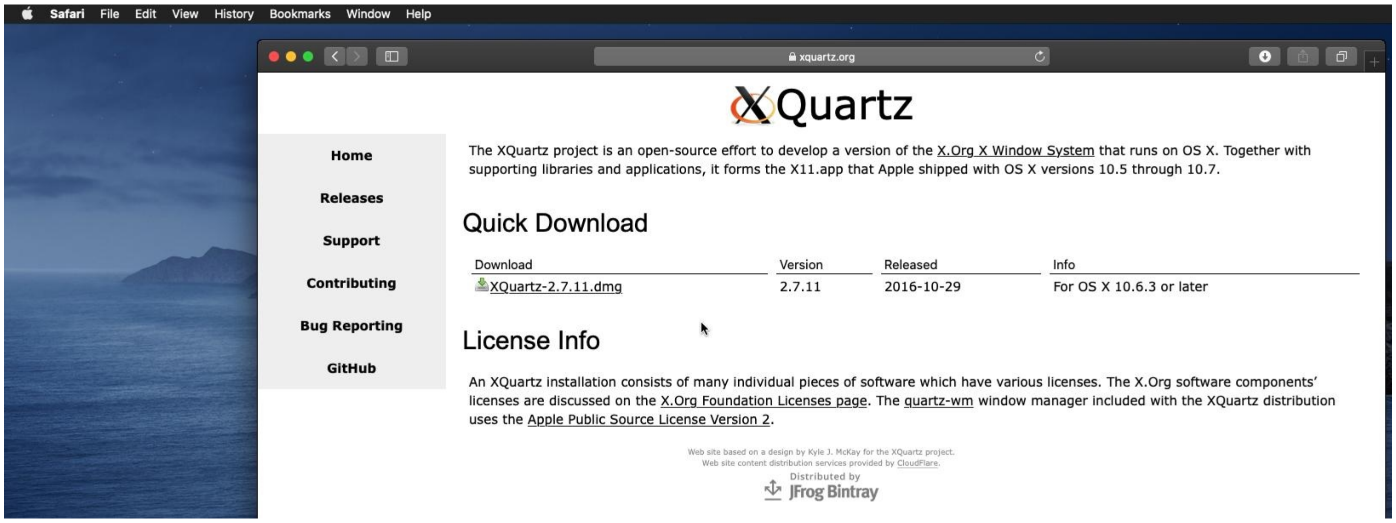
```
export DISPLAY=172.16.1.186:0.0
```

That command set the X-Windows display to be the address of the Windows machine and since Xlaunch is running, we have an active X-Display. The next command is:

```
ssh -X scott@172.16.1.225 nautilus
```

The 172.16.1.225 address is an Ubuntu desktop machine. The "-X" directive means to forward the X-11 traffic back to the current display we just set. Nautilus is the default Ubuntu file manager and you are seeing the nautilus file manager from the ubuntu machine displayed as a Window in Microsoft Windows 10.

On MacOSX, we can also run an X-Server. Go over to xquart.org and download the xQuartz server.



The XQuartz project is an open-source effort to develop a version of the X.Org X Window System that runs on OS X. Together with supporting libraries and applications, it forms the X11.app that Apple shipped with OS X versions 10.5 through 10.7.

Quick Download

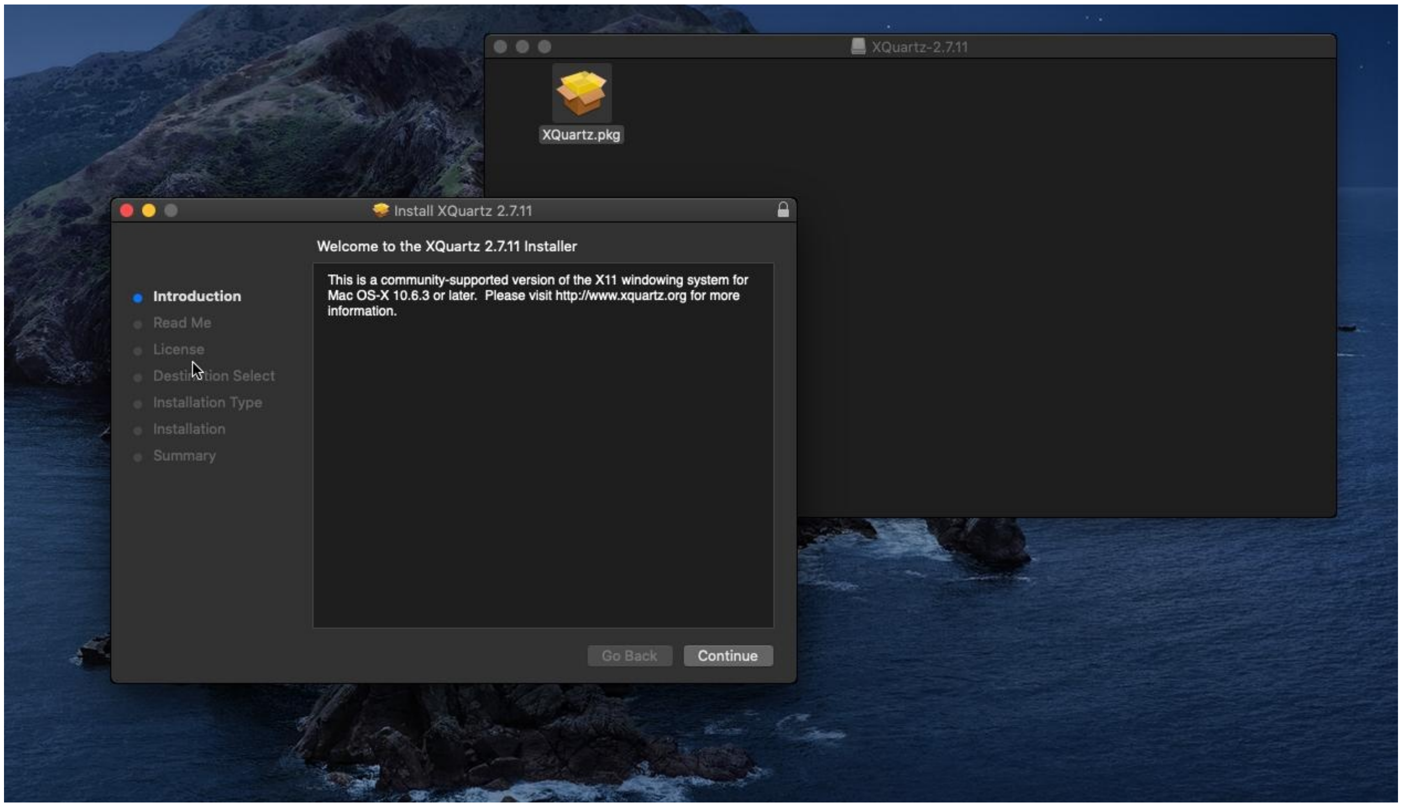
Download	Version	Released	Info
XQuartz-2.7.11.dmg	2.7.11	2016-10-29	For OS X 10.6.3 or later

License Info

An XQuartz installation consists of many individual pieces of software which have various licenses. The X.Org software components' licenses are discussed on the X.Org Foundation Licenses page. The quartz-wm window manager included with the XQuartz distribution uses the Apple Public Source License Version 2.

Web site based on a design by Kyle J. McKay for the XQuartz project.
Web site content distribution services provided by CloudFlare.
Distributed by
JFrog Bintray

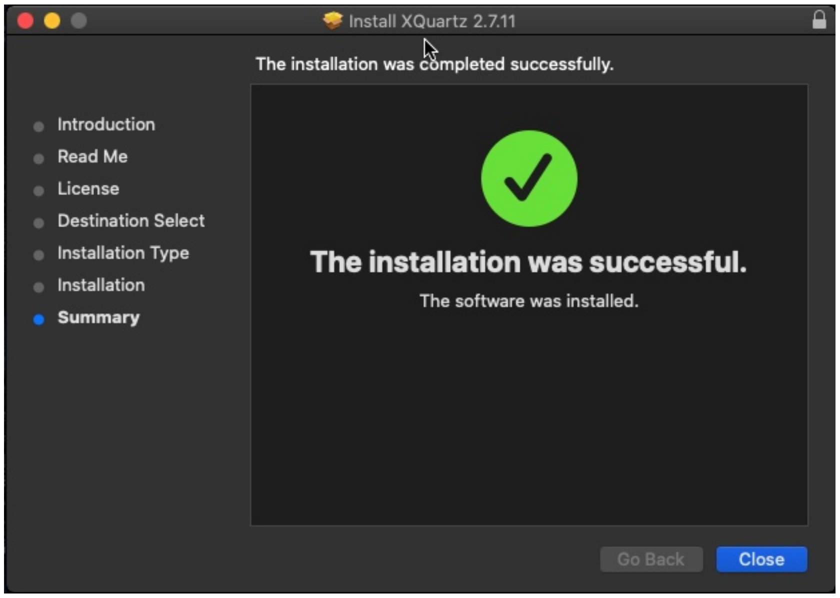
Launch the xQuartz installation package.



Welcome to the XQuartz 2.7.11 Installer

This is a community-supported version of the X11 windowing system for Mac OS-X 10.6.3 or later. Please visit http://www.xquartz.org for more information.

Complete the installation.

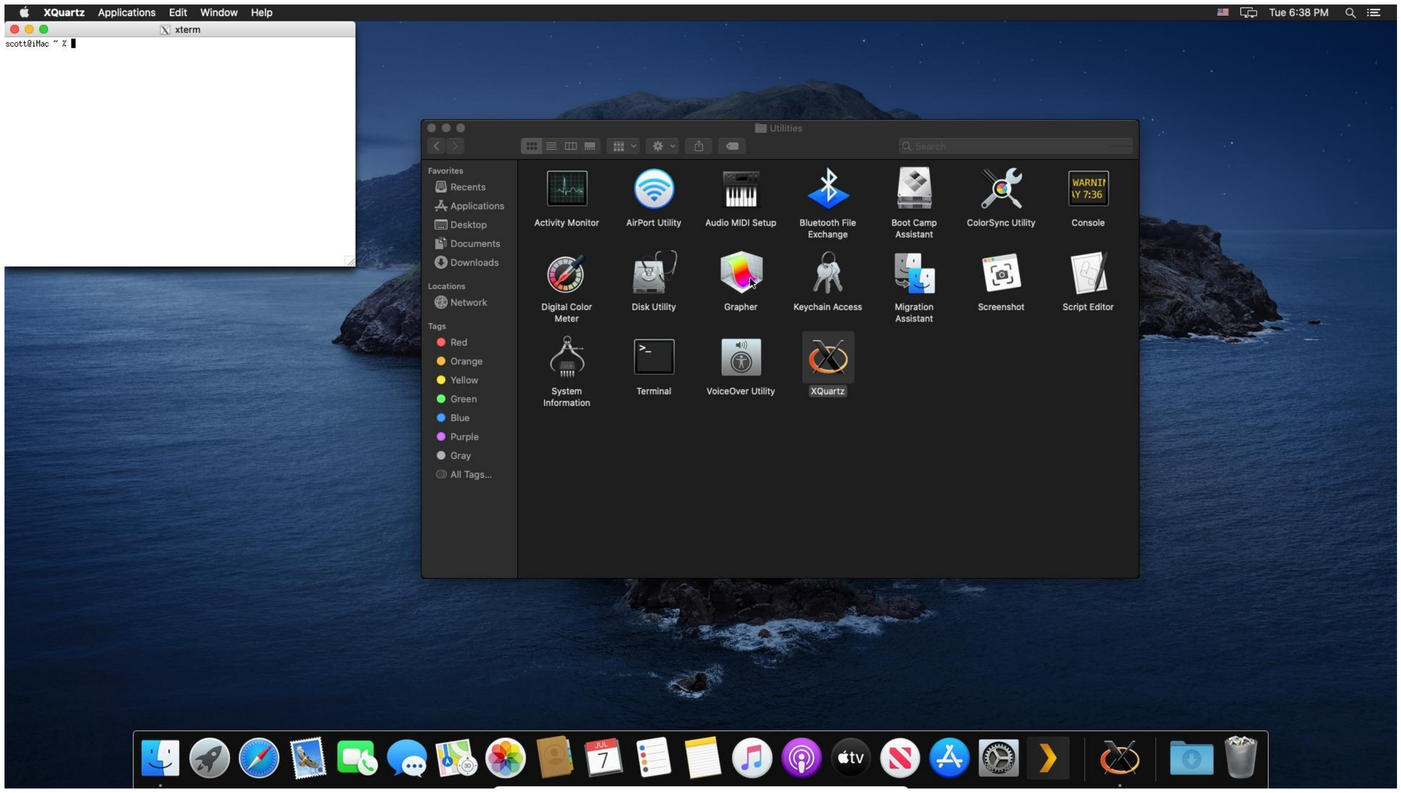


The installation was completed successfully.

The installation was successful.

The software was installed.

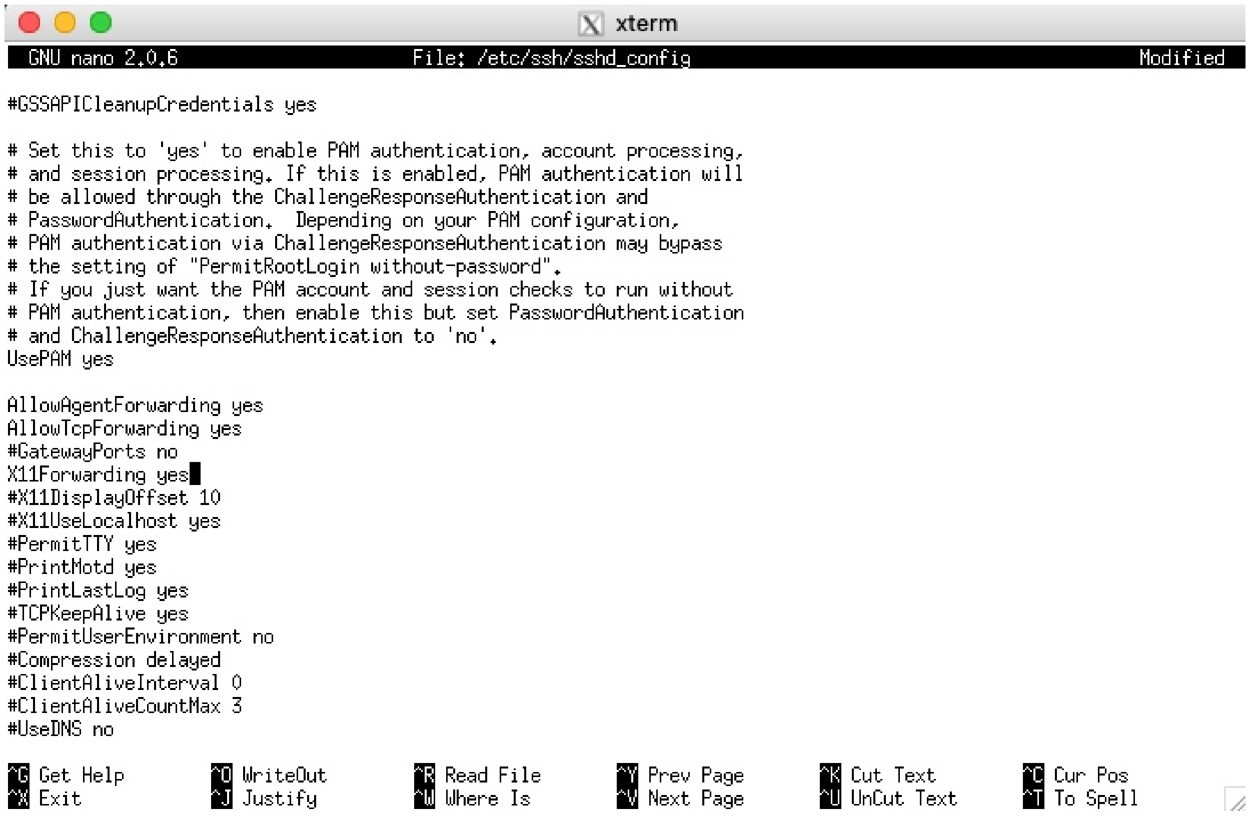
Launch XQuartz under Utilites in the Finder. Note that an Xterminal comes up.



In the terminal:

```
sudo nano /etc/ssh/sshd_config
```

Make the settings in the following screenshot:

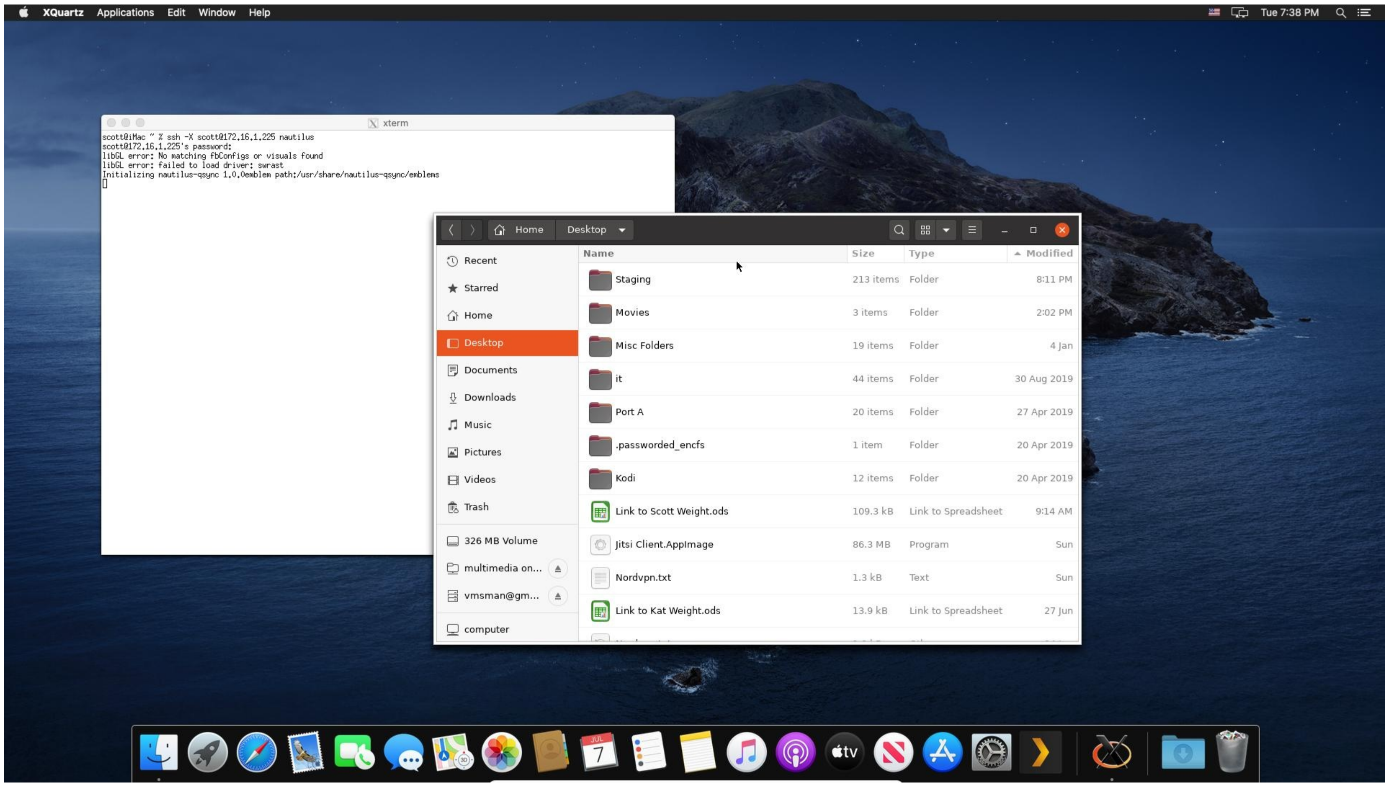


```
  GNU nano 2.0.6               File: /etc/ssh/sshd_config                    Modified

#GSSAPICleanupCredentials yes

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the ChallengeResponseAuthentication and
# PasswordAuthentication.  Depending on your PAM configuration,
# PAM authentication via ChallengeResponseAuthentication may bypass
# the setting of "PermitRootLogin without-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and ChallengeResponseAuthentication to 'no'.
UsePAM yes

AllowAgentForwarding yes
AllowTcpForwarding yes
#GatewayPorts no
X11Forwarding yes
#X11DisplayOffset 10
#X11UseLocalhost yes
#PermitTTY yes
#PrintMotd yes
#PrintLastLog yes
#TCPKeepAlive yes
#PermitUserEnvironment no
#Compression delayed
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS no

^G Get Help     ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit         ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

Save the file, restart the Mac and run Xlaunch after you login. I used the terminal ssh command:

```
ssh -X scott@172.16.1.225 nautilus
```

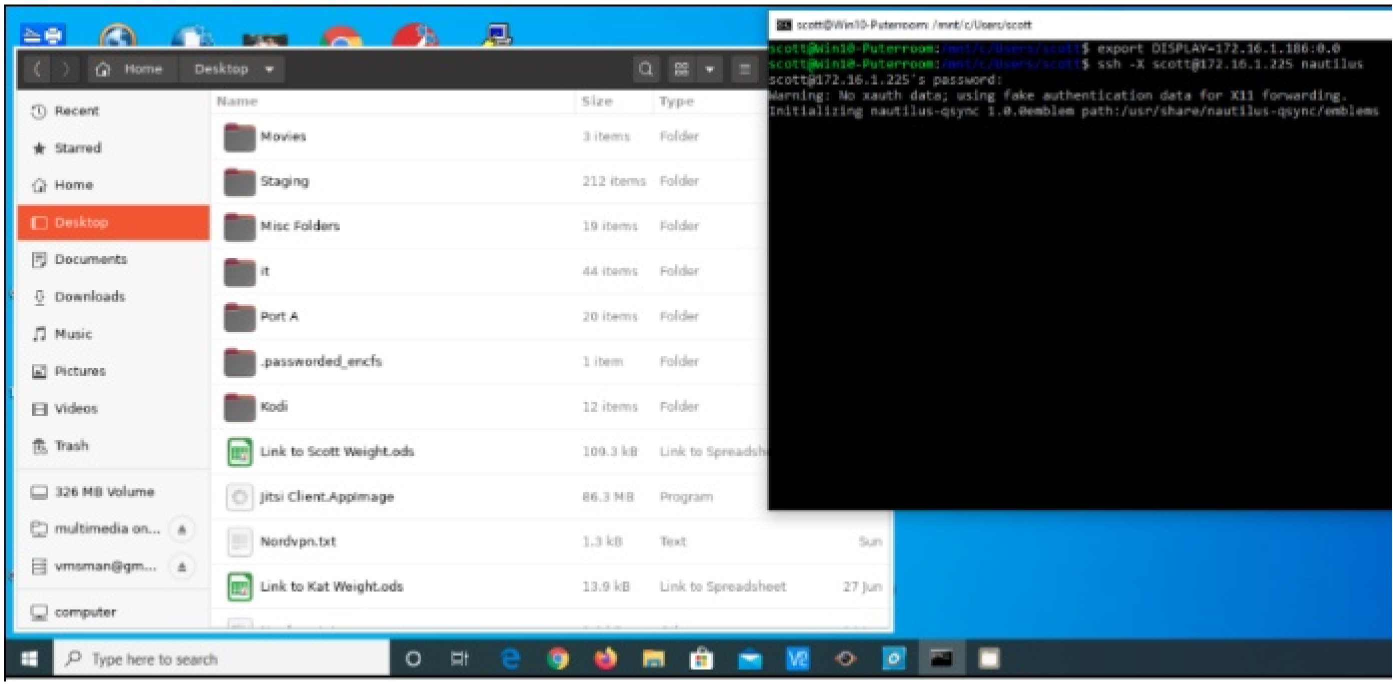
My Ubuntu desktop is running at 172.16.1.225 and the above command launched the nautilus file manager and displayed it back on the Mac running the XQuartz X-11 server at 172.16.1.136:



```
scott@iMac ~ % ssh -X scott@172.16.1.225 nautilus
scott@172.16.1.225's password:
libGL error: No matching fbConfigs or visuals found
libGL error: failed to load driver: swrast
Initializing nautilus-qsync 1.0.0emblem path:/usr/share/nautilus-qsync/emblems
```

Name	Size	Type	Modified
Staging	213 items	Folder	8:11 PM
Movies	3 items	Folder	2:02 PM
Misc Folders	19 items	Folder	4 Jan
it	44 items	Folder	30 Aug 2019
Port A	20 items	Folder	27 Apr 2019
.passworded_encfs	1 item	Folder	20 Apr 2019
Kodi	12 items	Folder	20 Apr 2019
Link to Scott Weight.ods	109.3 kB	Link to Spreadsheet	9:14 AM
Jitsi Client.AppImage	86.3 MB	Program	Sun
Nordvpn.txt	1.3 kB	Text	Sun
Link to Kat Weight.ods	13.9 kB	Link to Spreadsheet	27 Jun

This is logically the same as the Windows example of running the same nautilus screen off the Ubuntu host which we ran earlier.



```
scott@Win10-Puterroom:/mnt/c/Users/scott$ export DISPLAY=172.16.1.186:0.0
scott@Win10-Puterroom:/mnt/c/Users/scott$ ssh -X scott@172.16.1.225 nautilus
scott@172.16.1.225's password:
Warning: No xauth data; using fake authentication data for X11 forwarding.
Initializing nautilus-qsync 1.0.0emblem path:/usr/share/nautilus-qsync/emblems
```

Name	Size	Type	
Movies	3 items	Folder	
Staging	212 items	Folder	
Misc Folders	19 items	Folder	
it	44 items	Folder	
Port A	20 items	Folder	
.passworded_encfs	1 item	Folder	
Kodi	12 items	Folder	
Link to Scott Weight.ods	109.3 kB	Link to Spreadsh	
Jitsi Client.AppImage	86.3 MB	Program	
Nordvpn.txt	1.3 kB	Text	Sun
Link to Kat Weight.ods	13.9 kB	Link to Spreadsheet	27 Jun

In this tutorial, we learned that the "-X" option on the "ssh" command can perform X-Windows forwarding. We learned that neither Windows 10 or MacOSX Catalina have an X-11 server packaged with them.   The X-11 Server is an indigeneous part of all Linux computers that can be simply configured.

We can provide X-Server capability by adding software to Windows and Mac. Therefore, we installed and configured VcXsrv on Windows 10 and XQuartz on MacOSX in order to have X-11 servers capable of displaying X-Windows applications from both local and remote X-Windows client applications.

We configured the openssh-server on both Windows and Mac in order to accept inbound connections and made sure they could issue ssh -X forwardinig commands.

This allows for running GUI client applications which may be installed on virtual machines, docker containers, or LXC containers. Another useful application for this might be accessing and running applications from Raspberry Pi's and displaying those application Windows back to Windows 10, MacOSX Catalina or Linux desktop computers.