# This document describes the steps required to setup debug configuration on eclipse using Eclipse Remote System Explorer.

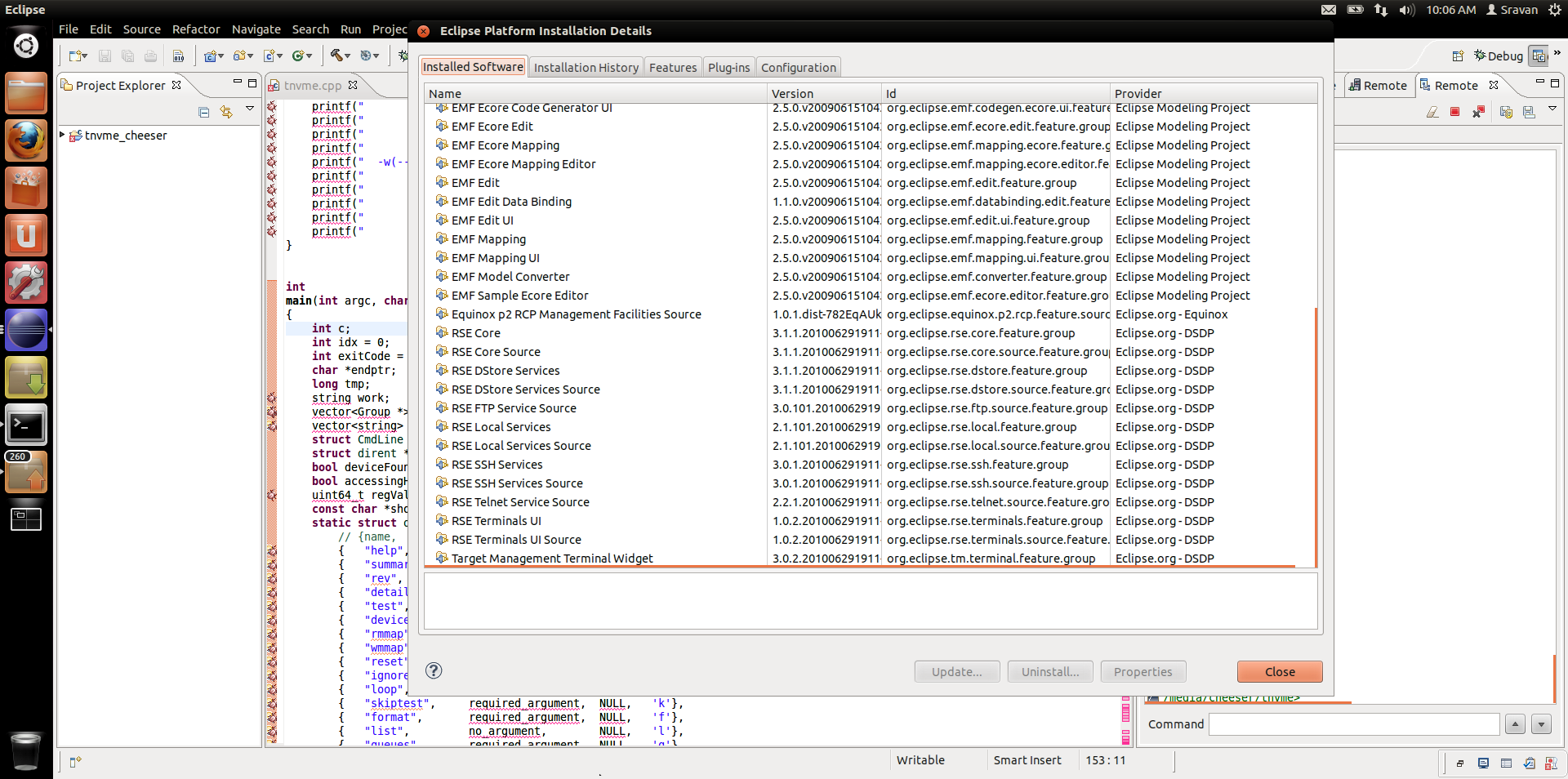
**Eclipse Remote System Explorer (RSE) Installation**

The Remote System Explorer (RSE) is a perspective and toolkit in Eclipse workbench, which allows you to connect and work with a variety of remote systems. With the predefined plug-ins, you can look at remote file systems, transfer files between hosts, do remote search, execute commands and work with processes.

On your Host system, you should have already installed Eclipse CDT (C/C++ development) if not then you will need it to proceed. Once you have your eclipse installed then launch eclipse on the host OS.

Host system is where you generally develop your application. It can be any linux system. For example you can launch Ubuntu from Windows VirtualBox and this ubuntu within the virtualbox is the host system or it can be a direct host with linux on it.

First step is to install the RSE using the eclipse menu option, Click on Help->Install New Software. Install the RSE and Target communication framework components in your preferred method. At the end of installation of RSE when you look at the installed software, you should see the below RSE components.



For more information on RSE follow the link <http://help.eclipse.org/helios/index.jsp?nav=%2F46>.

**Below are the steps for setting up network on QEMU’s Guest OS.**

You may have noticed that with default settings on QEMU, the guest operating system's network is assigned an internal network address, usually 10.0.2.15, and this is not accessible from the outside. This following steps allows the guest OS (linux) to connect to other other machines using a bridge connection.

On your Host OS (here Host is the linux server, not the VirtualBox Linux system):

First the network interface must be setup on the host OS. For Ubuntu, the file is located at /etc/network/interfaces. To set it up to act as a bridge, change the file to look like this (assuming eth0 is the physical ethernet port you want to use as a bridge).

Entry /etc/network/interfaces: For DHCP configured Host:

# This file describes the network interfaces available on your system

# and how to activate them. For more information, see interfaces(5).

# The loopback network interface

auto lo

iface lo inet loopback

# The primary network interface

auto eth0

iface eth0 inet dhcp

# The bridge network interface(s)

auto br0

iface br0 inet dhcp

bridge\_ports eth0

bridge\_fd 9

bridge\_hello 2

bridge\_maxage 12

bridge\_stp off

Note: Change all eth0 to eth1 if your network interface uses eth1

Once you set this configuration file up, you need to restart the network (sudo /etc/init.d/networking restart), or restart the system. Next, we need to specify a script for qemu to run when it wants to 'tap' into the network and claim a vlan.

Open a terminal and edit the file /etc/qemu-ifup. Comment out the existing lines in this file and add the below lines.

#sudo vim /etc/qemu-ifup

#!/bin/sh

echo "Executing /etc/qemu-ifup"

echo "Bringing up $1 for bridged mode..."

sudo /sbin/ifconfig $1 0.0.0.0 promisc up

echo "Adding $1 to br0..."

sudo /usr/sbin/brctl addif br0 $1

sleep 2

Then commented out lines should be as below

#switch=$(/sbin/ip route list | awk '/^default / { print $5 }')

#echo "Switch " ${switch} "Param " $1

#/sbin/ifconfig $1 0.0.0.0 up

#/usr/sbin/brctl addif ${switch} $1

In your host OS, open nvme/manage/runQemu.sh script and edit the line launching the QEMU to add the above network nic, vlan and tap: (no spaces between commas).

sudo ../qemu/x86\_64-softmmu/qemu-system-x86\_64 -m 4096 -hda ./vdisk/${vdisk} -device nvme,namespaces=2,size=512 --enable-kvm -smp 2 -net nic,vlan=0 –net tap,vlan=0,ifname=tap0,script=/etc/qemu-ifup

Once you launch QEMU guest OS, verify if the Guest OS has IP assigned with dynamic IP.

$ ifconfig

Ping this IP from your virtualBox ubuntu and check if your QEMU guest OS is visible to VirtualBox guest Linix system. If it successfully pings then you are all set for next section. If any issues with the network setup refer the online documentation for your specific errors.

For more information on QEMU networking, below are the links

<http://www.h7.dion.ne.jp/~qemu-win/HowToNetwork-en.html>

<http://people.gnome.org/~markmc/qemu-networking.html>

For Intel users more information can be found at

<http://dcgshare.lm.intel.com/wiki/QemuEmulation#Advanced_Networking>

**Starting the RSE server on QEMU Guest OS.**

Download the RSE server available at the below link to your QEMU guest OS.

<http://www.eclipse.org/downloads/download.php?file=/tm/downloads/drops/R-3.3.2-201202061000/rseserver-3.3.2-linux.tar>

If the above location does not work then google for the latest version of rseserver available from ecliplse.org/downloads.

You can start the RSE communications server with the server daemon, or manually. Before starting the server, make sure the Java command is in your path, you can do this by running the following command:

java -version

You should see something similar to the following:

java version "1.6.0\_20"

Etc.,

Also ensure that QEMU guest OS has perl installed on.

In a terminal on QEMU guest OS

$ sudo mkdir /opt/rseserver

$ chmod 777 /opt/rseserver

Copy rseserver-3.3.2-linux.tar to /opt/rseserver and in /opt/rseserver directory,

$ tar –xvf rseserver-3.3.2-linux.tar

$ sudo –l root

$ cd /opt/rseserver

$ perl ./daemon.pl

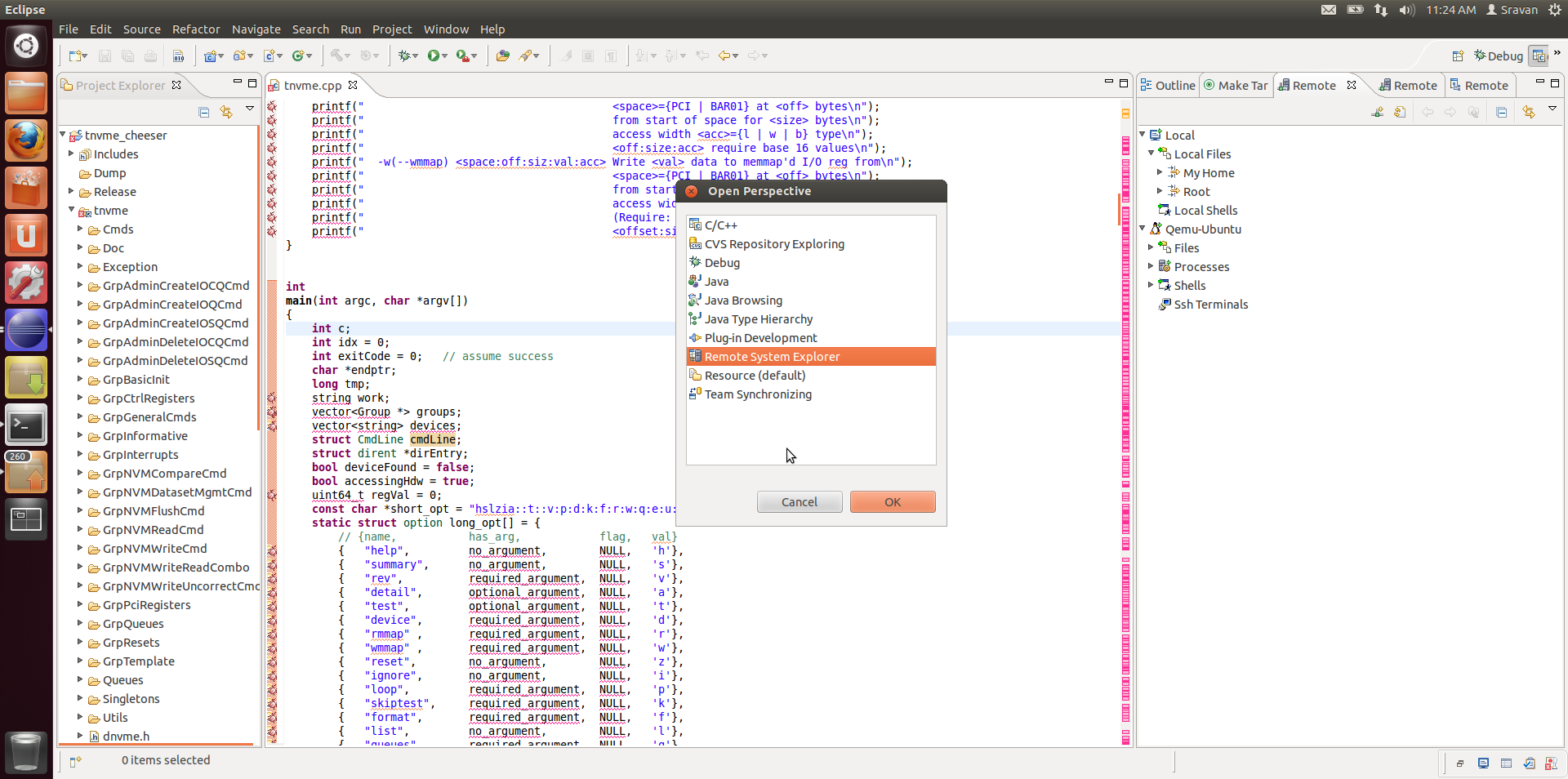
For more information please refer to the below link

<http://www.eclipse.org/ptp/documentation/3.0/org.eclipse.ptp.rdt.doc.user/html/gettingstarted/server_installation_unix.html>

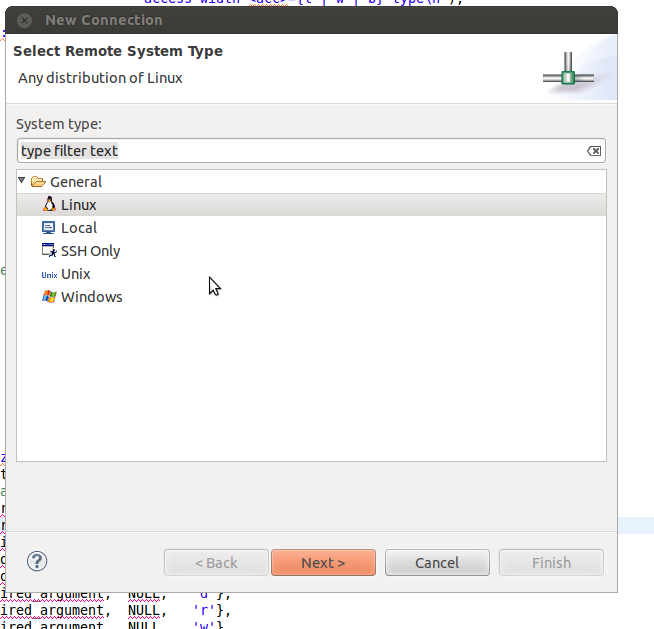
**Steps to configure eclipse to use Remote System Explorer**

On your Host system (Virtual Box Linux System or any Linux)

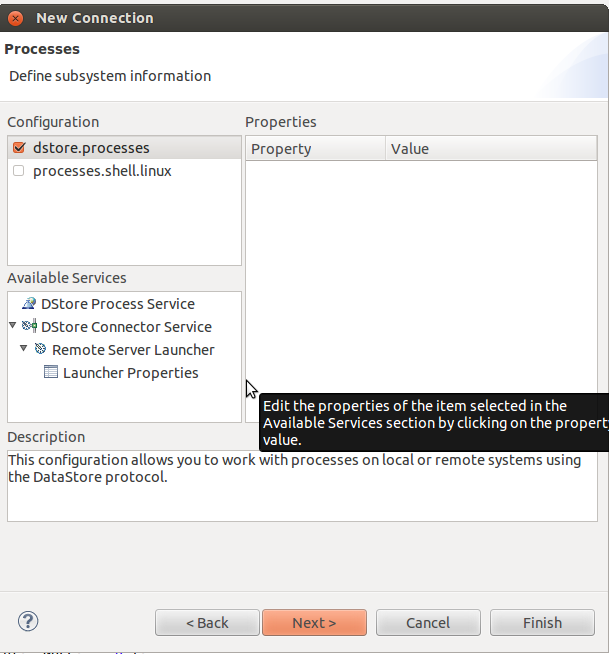
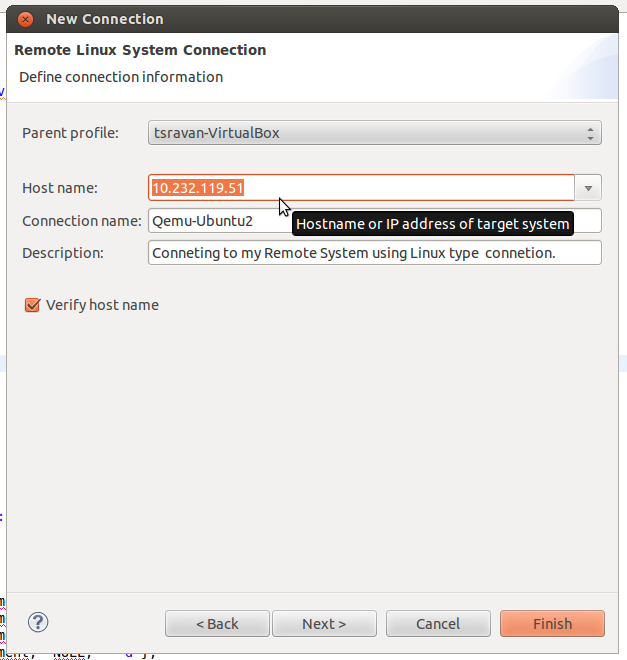
Add the RSE perspective to your workspace, Go to Window->Open perspective->Other, Select Remote System Explorer.

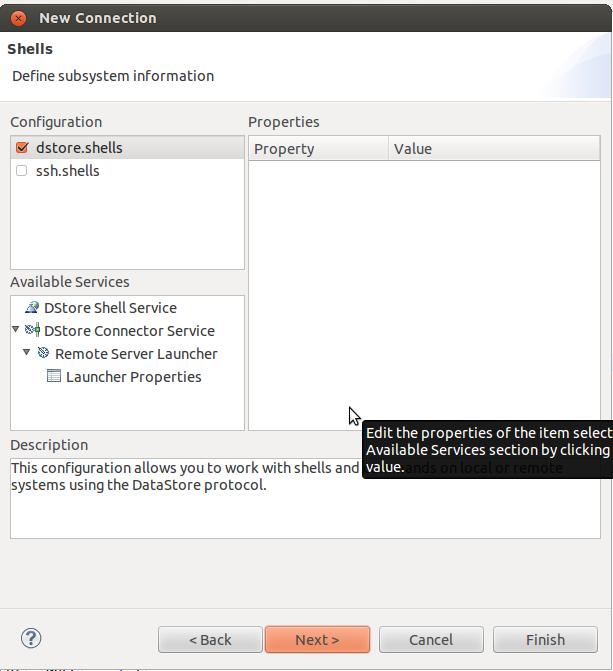
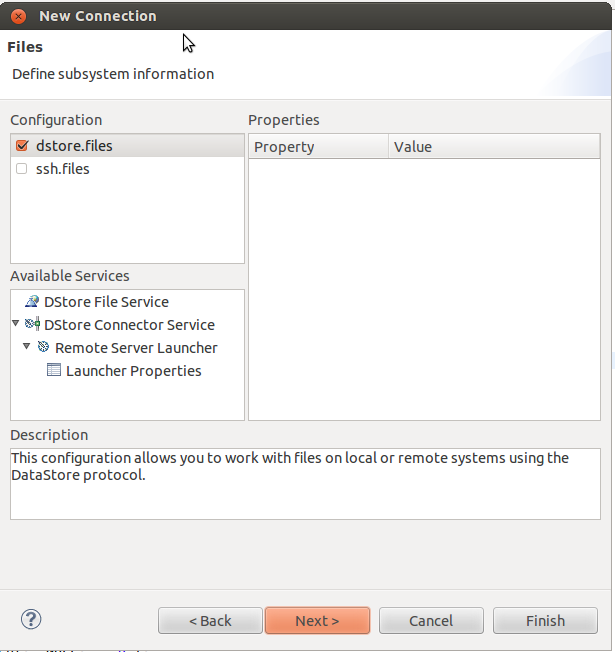


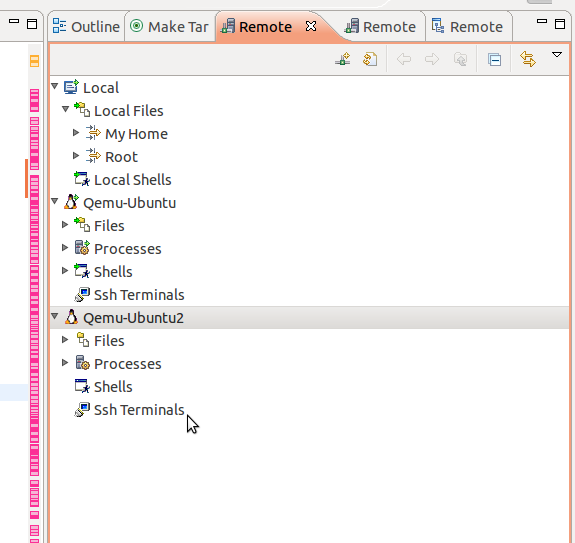
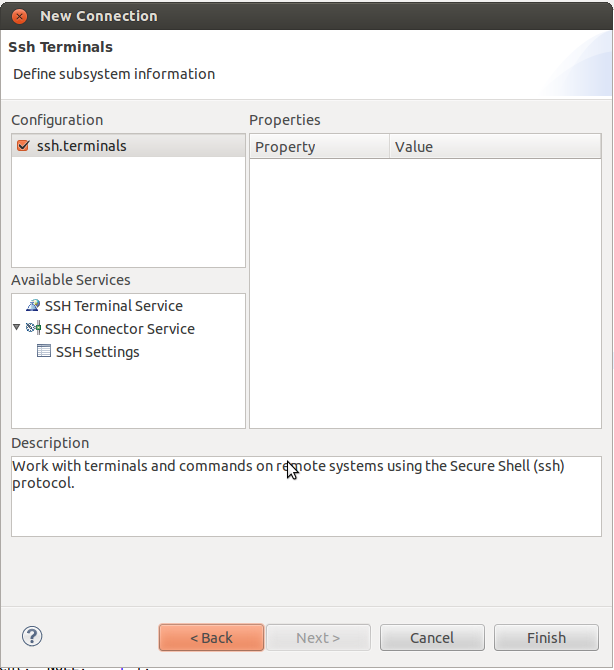
Now you will have a new tab in your Eclipse workspace named remote systems. Right click in this new Remote Systems tab and select “New Connection” which brings you to a new window below, where you will select Linux and click next.



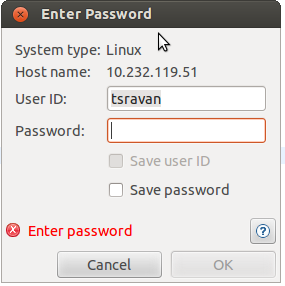
The next screen window appears where you specify the host IP or host name and your preferred connection name with optional description. Make sure that the Verify host name is checked. If the connection to remote is not established, it will not proceed and you will need to refer to the above sections of this document to establish a connection to QEMU Guest OS. You can either say finish or next and retain the default options. Some screenshots are shown to guide you through this process.





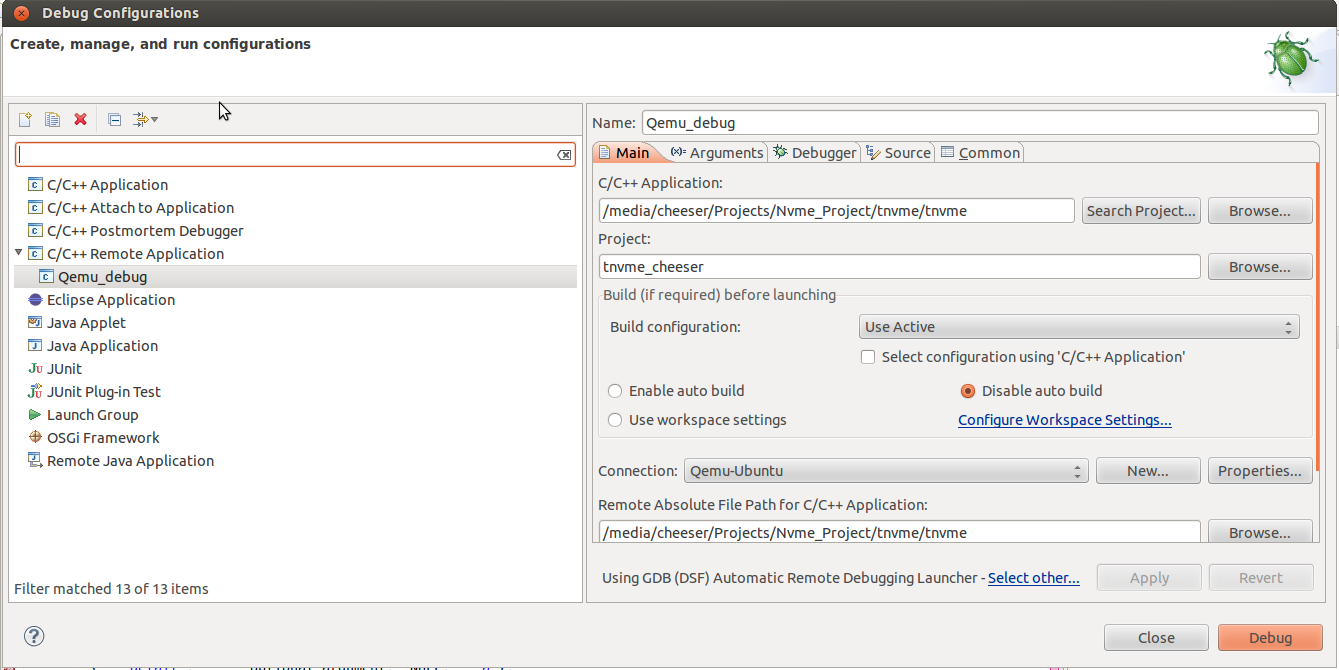


Once you are done you will see the New Connection specified in Remote Systems tab. Once you expand Files tab it will prompt you to enter user name and password of your remote guest OS.



**Now you are all set for a debug configuration**

Goto Run->DebugConfiguration and you will see the below dialog window. Where you select the C/C++ remote application and specify a name.

****

Make sure you have disable auto build radio button checked and specify the connection that you have created. Also specify the application paths approprioately.

The arguments tab allows you to specify arguments required for this application. Your application should be compiled with –g option in Makefile otherwise you will not be able to see the source or symbols.

Click on Debug button and you should see the debug session started.

If you get permission denied then in qemu guest OS type the below lines in a terminal

$ sudo chmod 777 /dev/nvme0

Note: If you are not using remote application then you can select C/C++ attach to application on your local system to debug a local application.