

Linux Based Git Usage and Operation Guide

Preface

Git is the popular free distributed version control tool, now it's been widely use in software development. Every git work folder is a standalone code base which having the full history record and version trace function.

This operation guide mainly introduces the daily used git operation commands according with the Linux environment setup, configurations and operations.

1. Linux Environment Setup

1.1 Ubuntu Installation

1.1.1 Download VirtualBox and Ubuntu Image

We recommend using the Ubuntu 10.04LTS and VirtualBox 3.1.8 which proved stable and no known issues, these image and executable are available on our ftp server in path of:

ftp://10.85.172.62/media/Ubuntu_VirtualBox

We can use the username "cifae" and password "cifae" to enter this directory by ftp tools such as leapFTP.

You can also input the following in your Internet Explorer to directly enter the directory

ftp://cifae:cifae@10.85.172.62/media/Ubuntu_VirtualBox

FTP directory `/media/Ubuntu_VirtualBox` at 10.85.172.62

To view this FTP site in Windows Explorer, click **Page**, and then click **Open FTP Site in Windows Explorer**.

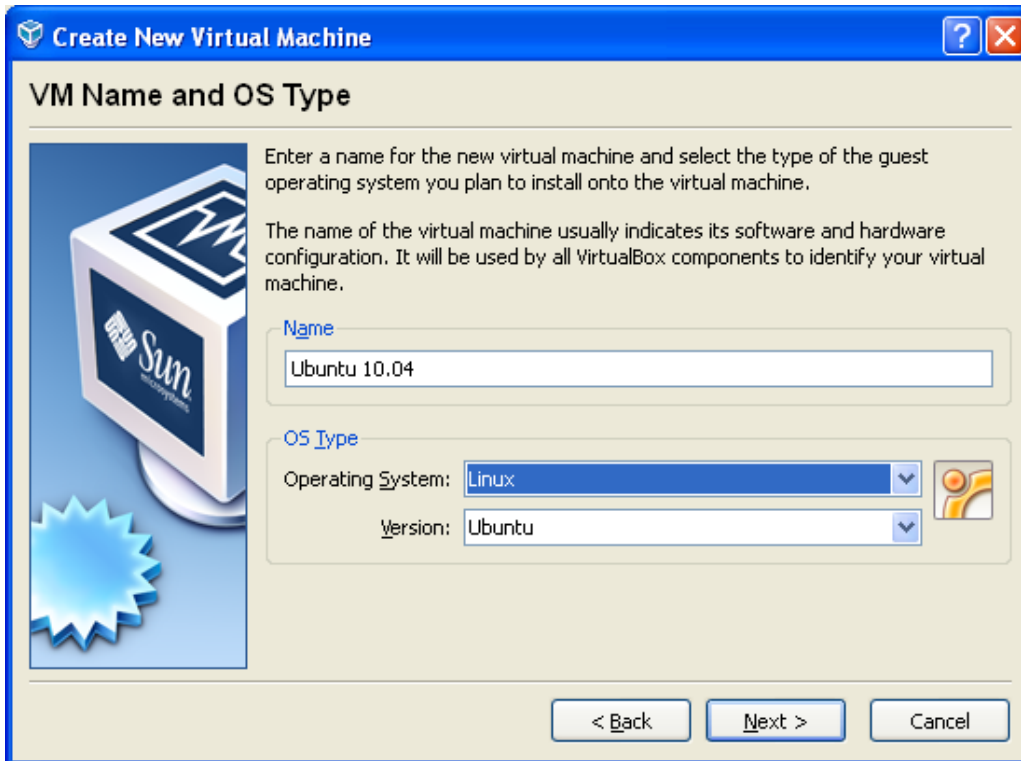
[Up to higher level directory](#)

05/15/2013 10:05AM	92,193,072	VirtualBox-4.1.8-75467-Win.exe
05/15/2013 10:10AM	58,707,718	gcc-linaro-arm-linux-gnueabi-2012.03-20120326_linux.tar.tar
05/15/2013 09:56AM	718,594,048	ubuntu-10.04.1-desktop-i386-ti.iso

1.1.2 Create the Virtual Machine with VirtualBox

- Run VirtualBox executable and install the software.
- Click the VirtualBox icon, and click "New" from the initial VirtualBox GUI.

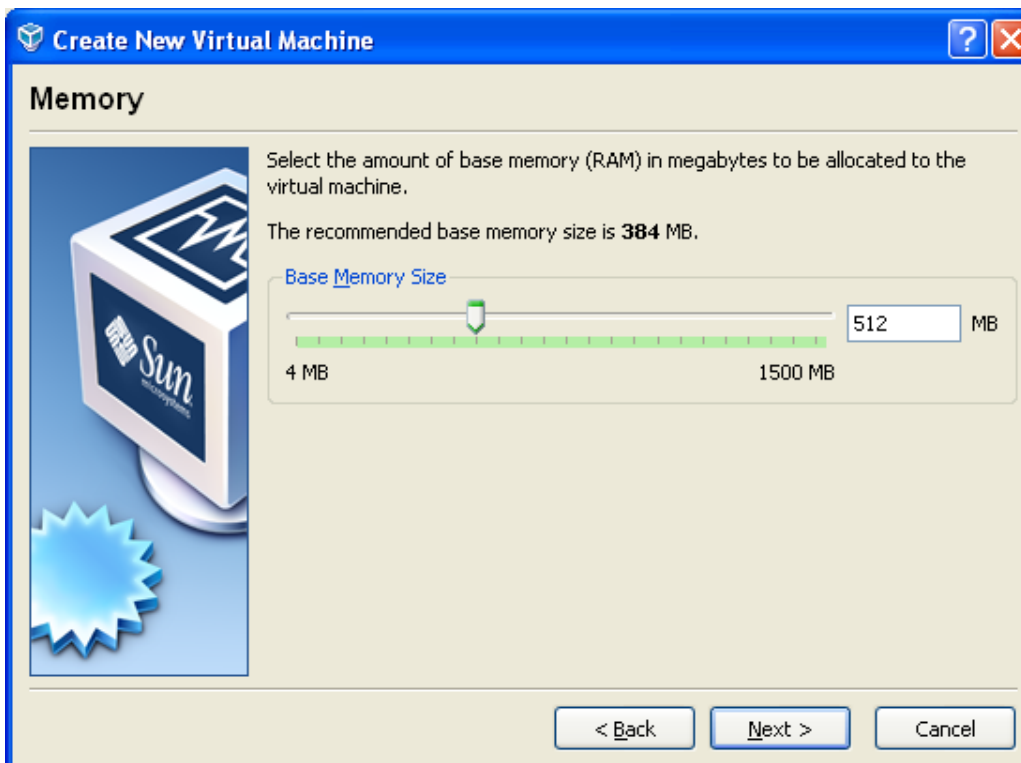
- This will launch a wizard that will step through the creation of the virtual machine. Click “Next”.
- Type a unique name for the new machine. A good name will include 'Ubuntu' and the Ubuntu version and maybe some other distinguishing info about the machine itself. This will help distinguish the virtual machine from others that may also be under VirtualBox.



The screenshot shows the 'Create New Virtual Machine' window with the title bar 'Create New Virtual Machine'. The main heading is 'VM Name and OS Type'. On the left is a graphic of a Sun Microsystems monitor. The text says: 'Enter a name for the new virtual machine and select the type of the guest operating system you plan to install onto the virtual machine. The name of the virtual machine usually indicates its software and hardware configuration. It will be used by all VirtualBox components to identify your virtual machine.'

There are two input fields: 'Name' with the text 'Ubuntu 10.04' and 'OS Type' with a dropdown menu set to 'Linux' and a sub-dropdown set to 'Ubuntu'. To the right of the OS Type dropdown is a small icon of a person. At the bottom are three buttons: '< Back', 'Next >', and 'Cancel'.

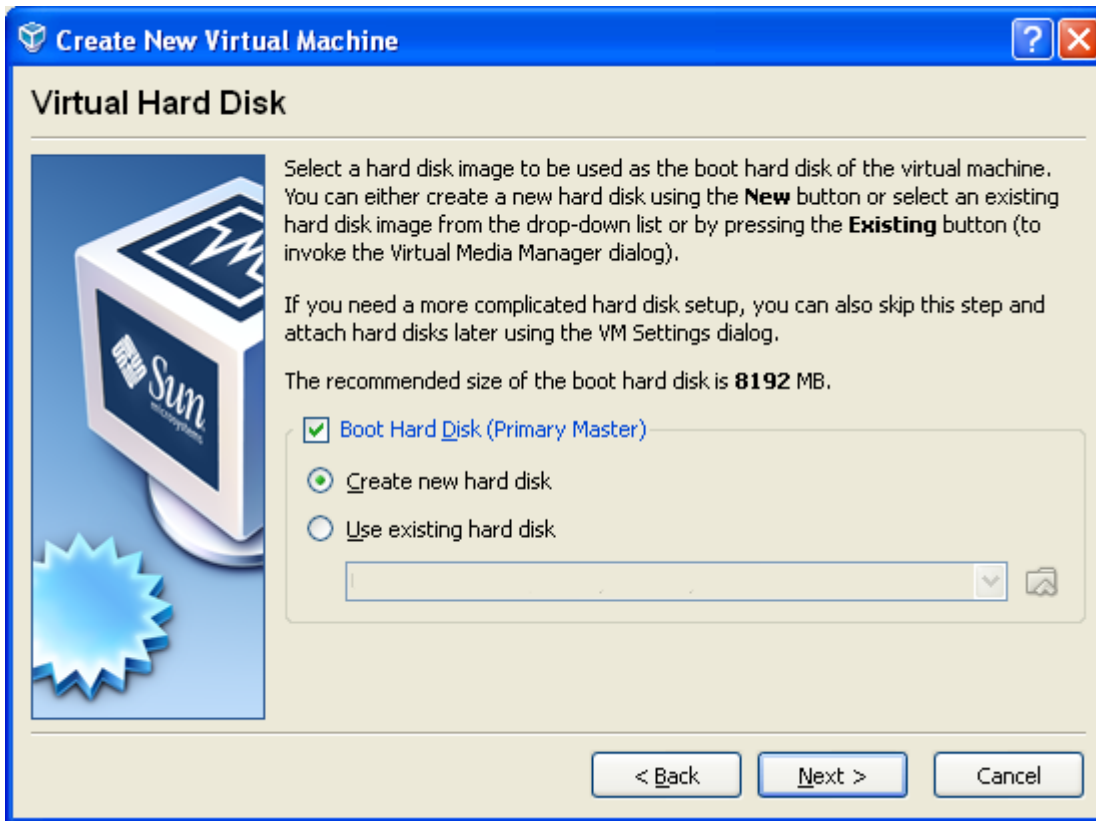
- You may want to increase the default RAM size. We have selected 512MB. Click “Next”.



The screenshot shows the 'Create New Virtual Machine' window with the title bar 'Create New Virtual Machine'. The main heading is 'Memory'. On the left is the same Sun Microsystems monitor graphic. The text says: 'Select the amount of base memory (RAM) in megabytes to be allocated to the virtual machine. The recommended base memory size is 384 MB.'

There is a slider for 'Base Memory Size' ranging from 4 MB to 1500 MB. The slider is set to 512 MB, which is also shown in a text box to the right of the slider. At the bottom are three buttons: '< Back', 'Next >', and 'Cancel'.

- Keep the default virtual hard disk settings (Create new Boot hard disk). Click “Next”.



- Select “Dynamically expanding storage” for the hard disk storage type. Click “Next”.

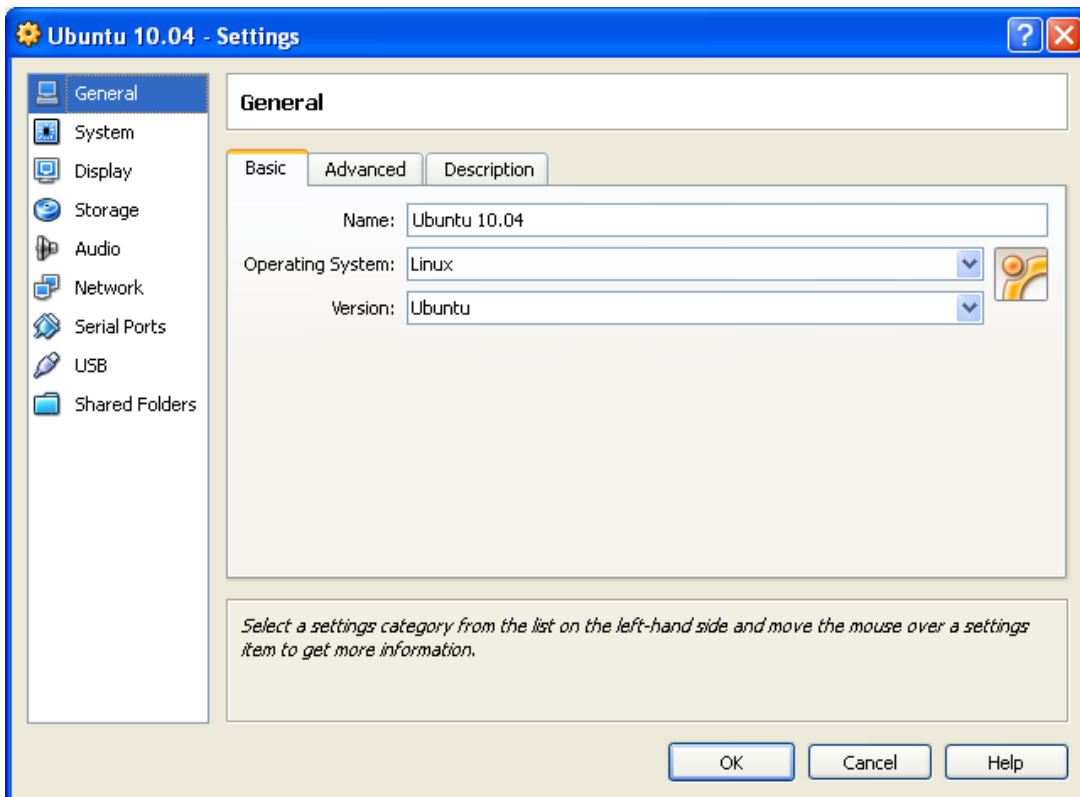
NOTE: It is highly recommended to select a hard disk of at least 20GB or higher. Keep in mind that this is the maximum size that your virtual hard drive can reach if you've selected the "dynamically expanding storage" option, i.e. it will not immediately take up that many GB of your hard drive. You will need this space for the SDK installation and your development.

- Click “Finish”.

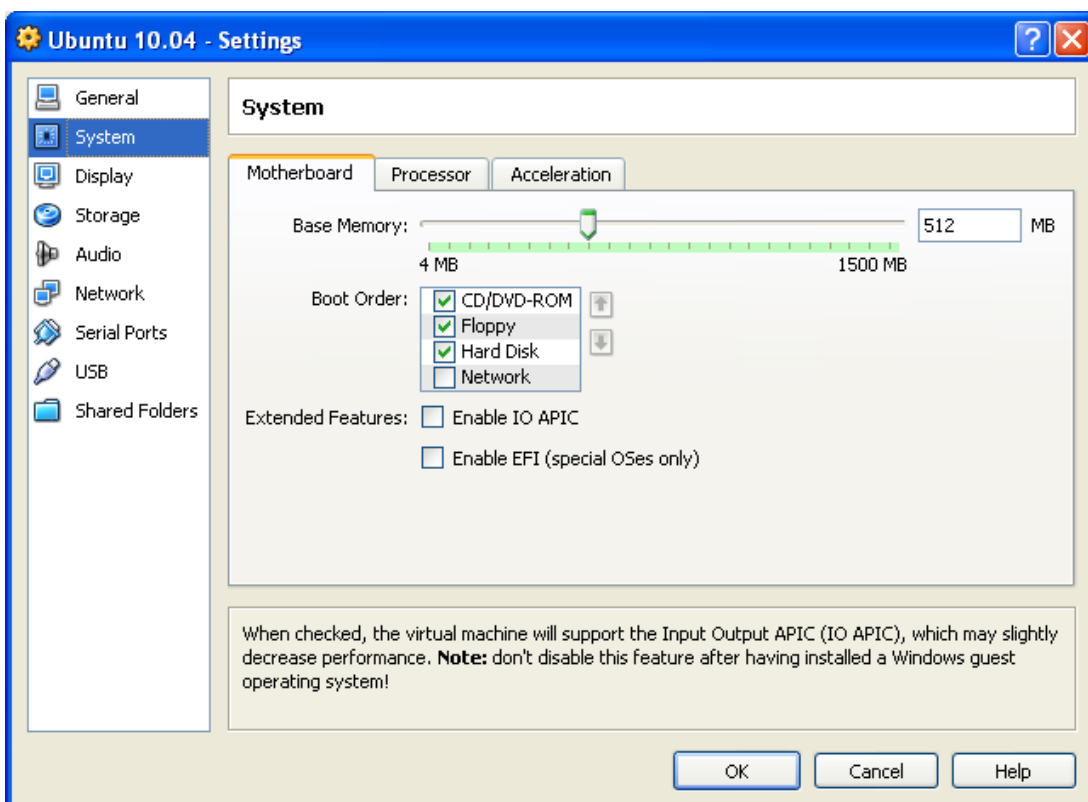
1.1.3 Configure Virtual Machine under VirtualBox

With your virtual machine now created in VirtualBox you need to configure your new machine by highlighting your machine name and clicking on Settings.

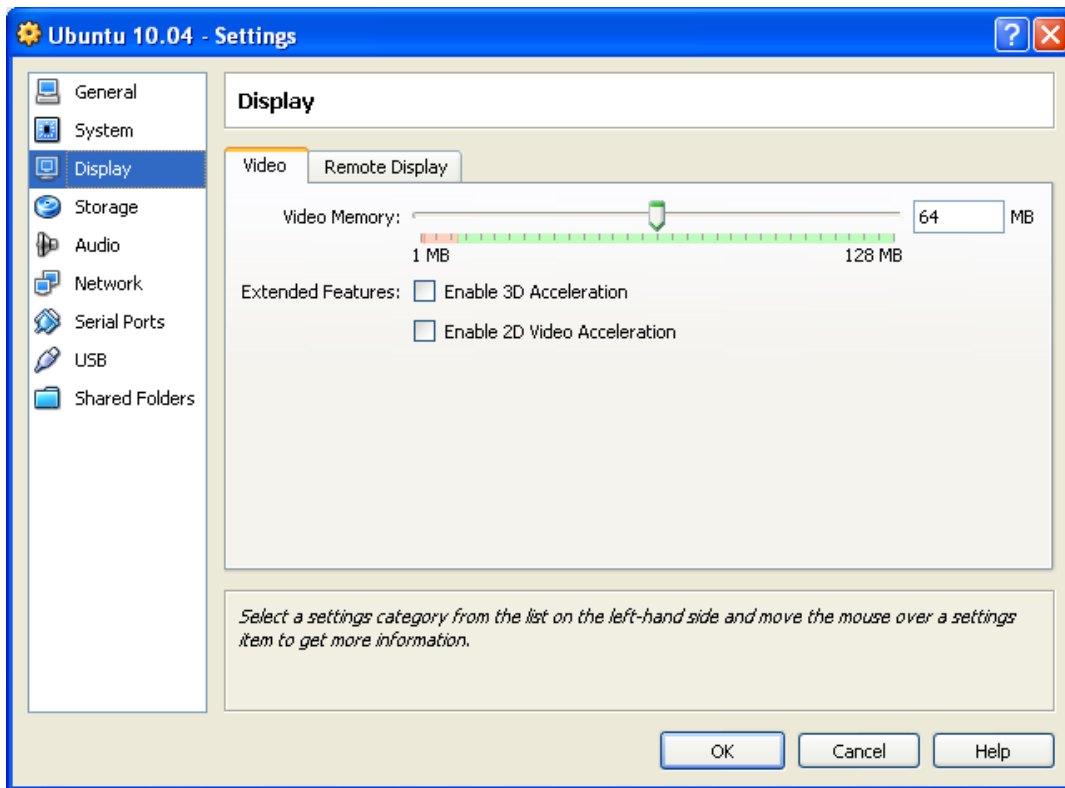
- Modify the OS Type. Set OS to Linux and Version to Ubuntu. Click “Next”.



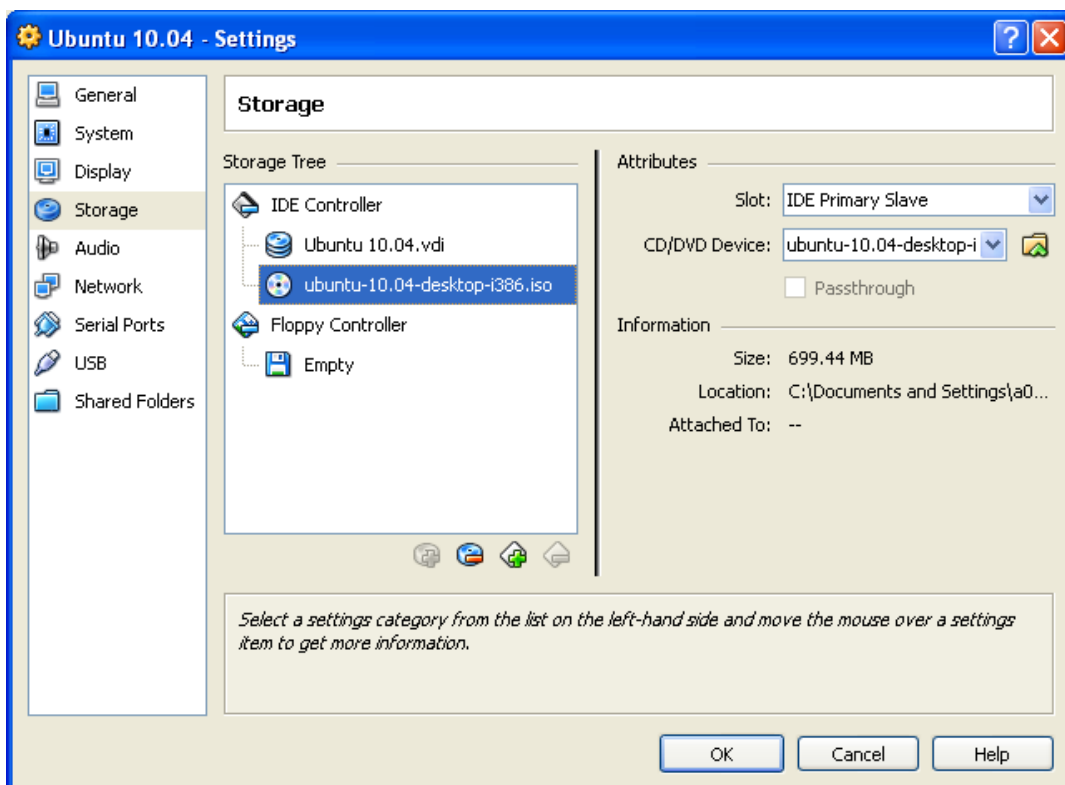
- If you have not already done so, you may want to increase the default RAM size of your VirtualBox. We have selected 512MB. Also set CD/DVD-ROM to highest Boot Order. Click Next.
- Note: Recommended RAM size is 1024MB -- else you may see swap and memory usage complete full causing builds to fail.



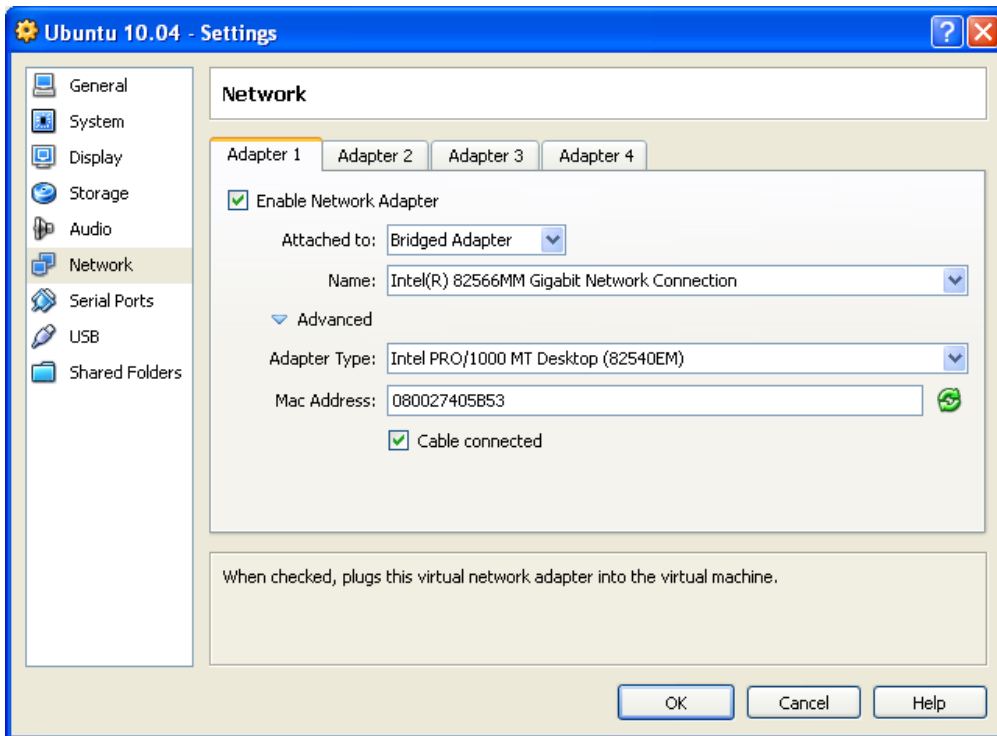
- It may be useful to change the default Video Memory to something more than 12MB. We have selected 64MB.



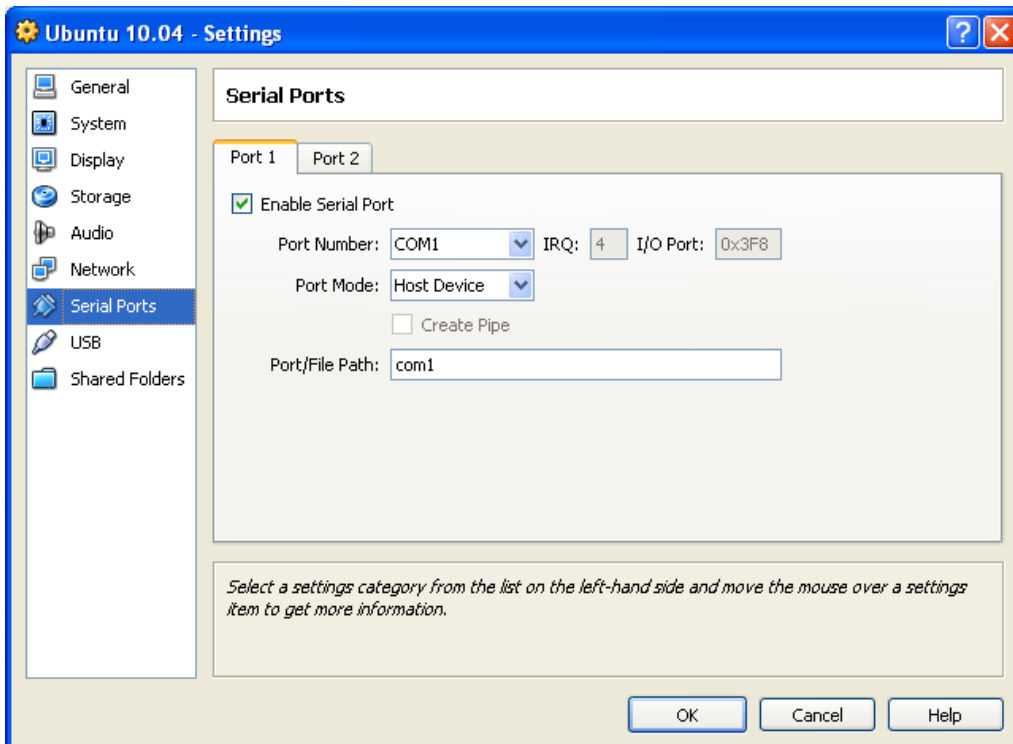
- Next Add an Attachment to Add a CD/DVD Device and point it to the ubuntu-10.04-desktop-i386.iso. When you start your virtual machine for the first time, it should boot from the CD/DVD and install Ubuntu 10.04 from this iso file.



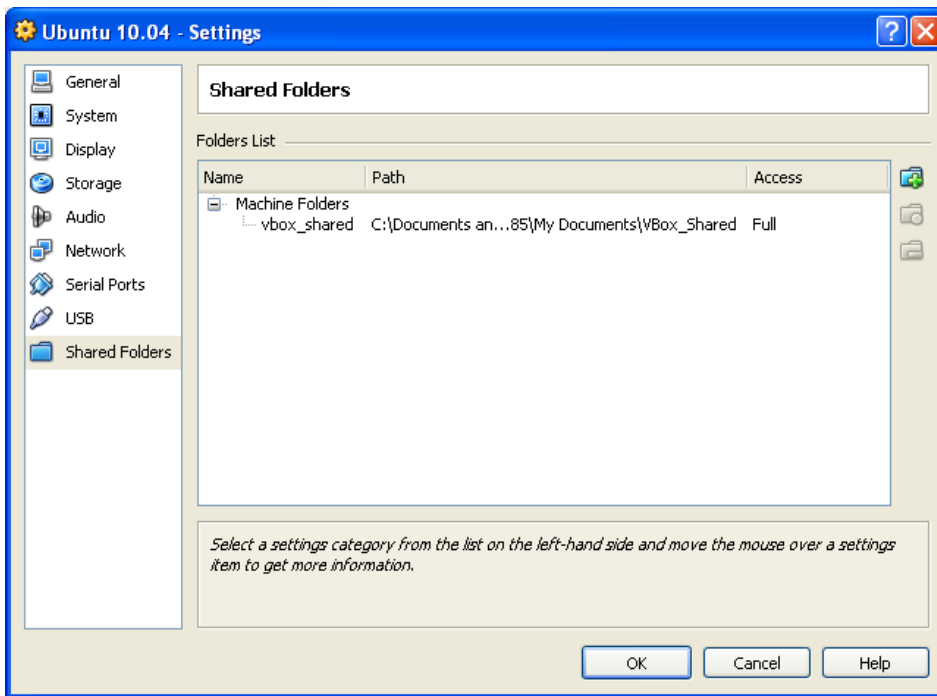
- Setting a Network Adapter will depend largely on your Windows machine and available network adapters. Enable Network Adapter and select Bridged Adapter for Attached To. If needed expand Advanced to see more adapters. We would not recommend using a Wireless adapter.



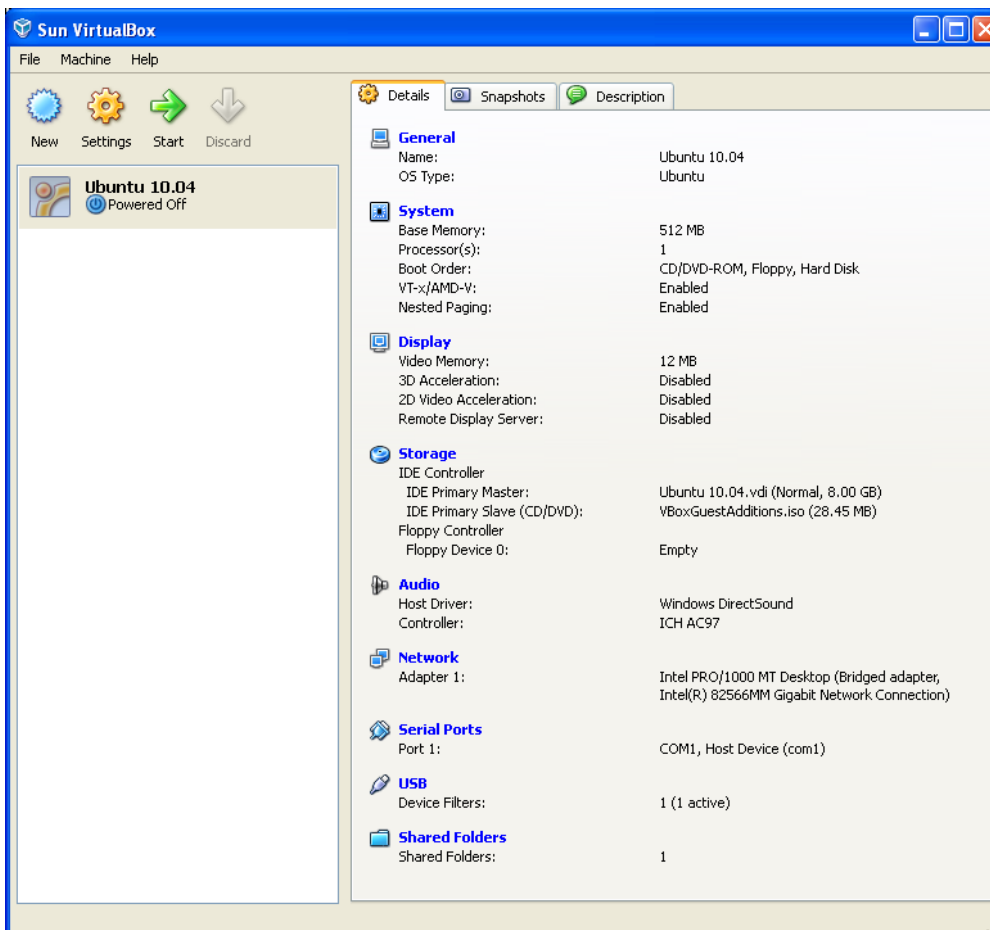
- You will need to enable a Serial Port to use minicom within your Linux machine. Enable the Serial Port, set the Port Number to COM1 (typical), Port Mode to Host Device and Port/File/Path to "com1".



- To enable sharing files between your Linux machine and Windows machine you can use the Share Folders feature of VirtualBox. For more information, see the later chapter.



When you are finished your VirtualBox should look similar to the screen below which shows all of the settings you configured for your Ubuntu Linux machine in the Details tab.



1.1.4 Install Ubuntu on the Virtual Machine

Once configured as seen above, your new virtual machine will display as "Powered Off". Click on the machine to highlight it and then click the Start button.

On Start your virtual machine should boot from CD/DVD and install Ubuntu 10.04 from the ISO file. The steps to install Ubuntu onto the virtual machine will vary depending on the version of Ubuntu. Follow the most reasonable steps to complete the installation. The steps will include the creation of a name for the machine. It can be (but doesn't have to be) the same as the name of the VirtualBox machine. It helps if the name designates the machine as Ubuntu with a version and a unique identifier. This name will be the hostname on a network.

The steps will also include the creation of a user account. This account will be able to process root access commands with the "sudo" command. A common username and password is shown below. These names are optional and your choice.

Username: *"user defined"*

Password: *"user defined"*

When installation completes, the machine will require a reboot. Since the ISO file has already installed Ubuntu it will ask you to Press Enter to remove the ISO file from the CD/DVD. Press Enter and the reboot will complete booting now from the virtual hard drive which runs Ubuntu.

1.2 Ubuntu Configuration

1.2.1 Install VirtualBox Guest Additions

With the virtual machine booted:

- From the VirtualBox Menu at the top of the window, Devices->Install Guest Additions.
- On the desktop you will notice a CD icon named "VBOXADDITIONS_x.x.x_xxxxx".
- Click it and run it, provide the user's password
- Upon installation, press "Enter", the installation window will close
- Right click on the CD, click "Eject"

Accept all the defaults. Guest Additions will allow the VirtualBox window to be resized. It also allows the mouse to move freely between the host and virtual machine and allows for cut/copy/paste between Windows host and virtual machine. Guest additions are also required for file sharing between the Windows and the Linux system.

1.2.2 Share Files between Ubuntu and Windows

VirtualBox has a Shared Folders feature which you need to setup before starting the Ubuntu image. In Virtual Box select your Ubuntu image and click Settings. At the bottom you will see Shared Folders. Specify (or create) the Windows folder you want to share with Ubuntu. Give the shared folder a name. It is a good idea to use the same folder name for both Windows and Ubuntu.

Using Shared Folders with VirtualBox requires that the VirtualBox Guest Additions has been installed. Also, Ubuntu updates often wreck the VirtualBox Guest Additions settings. If folder sharing stops working after Ubuntu update, reinstall Guest Additions.

Steps to share a folder between windows and Linux:

- Make sure the Virtual Machine is shut down.
- Create a folder on the windows machine, for example "c:\linux-share"
- Open the Oracle VM VirtualBox Manager
- Settings -> Shared Folders
- Add the folder "c:\linux-share" as shared folder. Use Auto-Mount=Yes; Access=Full
- Select "OK"
- Start the Virtual Machine and login.
- *From the VirtualBox Menu at the top of the window, Devices->Install Guest Additions.
- *On the desktop you will notice a CD icon named "VBOXADDITIONS_x.x.x_xxxx".
- *Click it and run it, provide the user's password
- *Upon installation, press "Enter", the installation window will close
- *Right click on the CD, click "Eject"
- Add your user id to "vboxsf" group: System->Administration->Users and Group->Manage Groups->vboxsf->Properties Check your username, OK, provide Password, Close
- Restart the Ubuntu 10.04
- Open a Terminal: Accessories->Terminal

- The shared folder is located in **/media/sf_linux-share**. You can use this directory for copying files between windows and VM Linux.
- Note that once a file is loaded to this folder it is owned by root and belongs to the group “vboxsf”. You will only be able to write to this file as “root”. However if you copy the file to your home directory (or any other directory that is user owned) the owner of the file will be the user. Perform `ls -al "filename"` to see the owner.

* The lines marked with * is optional when you haven't install “Guest Additions”

1.2.3 Configure Proxy in Ubuntu

If your network is behind a firewall you will need to configure the network proxy for Ubuntu in order to successfully download the applications required to complete your development environment or to browse the Internet on your Linux Host machine.

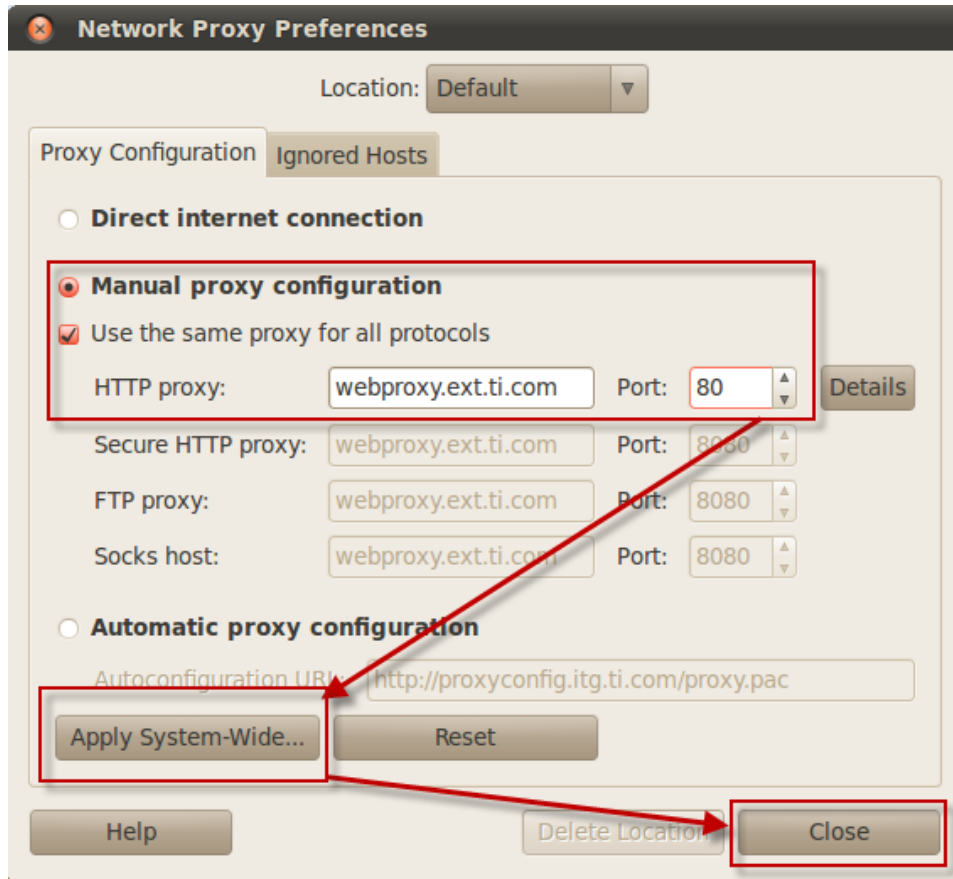
To configure the network proxy in Ubuntu, go to System->Preferences and click “Network Proxy” as seen below.



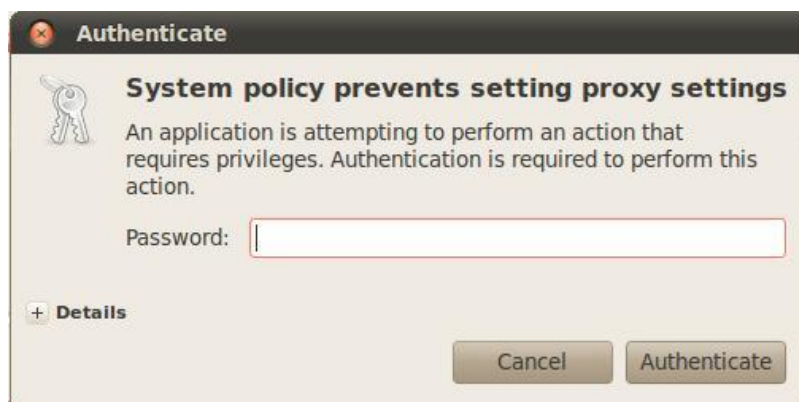
As seen in the image below, click “Manual Proxy Configuration”. Specify the HTTP proxy server used by your company. You may find this information under your Windows OS inside the Internet Explorer Network Connections. On your host machine, open Internet Explorer. Go to, ‘Tools’, ‘Internet Options’, in the ‘Connections’ tab, click ‘LAN settings’. Copy the proxy address shown. Be sure to specify the port.

Make sure you check “Use the same proxy for all protocols”. Also be sure to click “Apply System-Wide”.

The following snap shot is the configuration in TI Intranet.

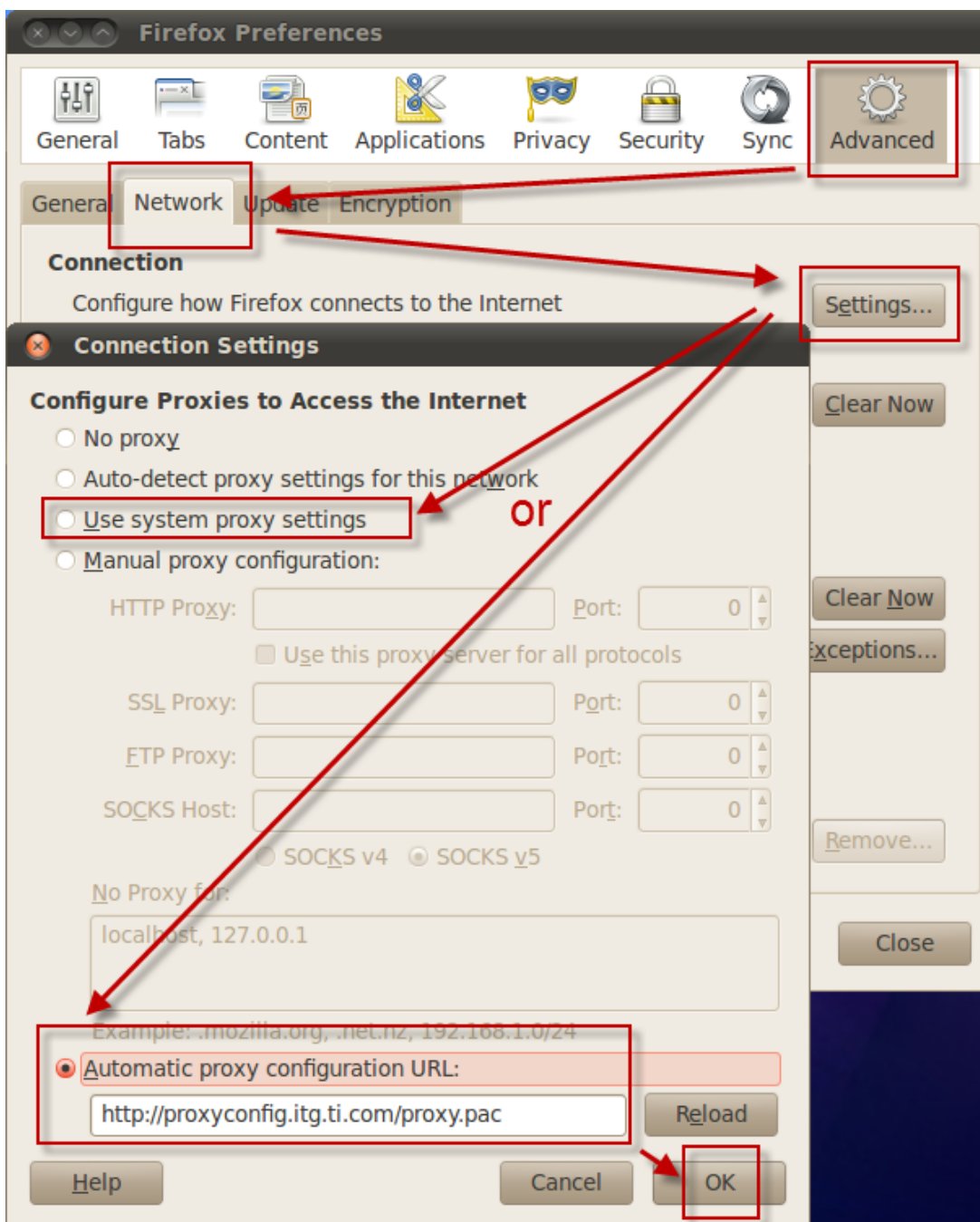
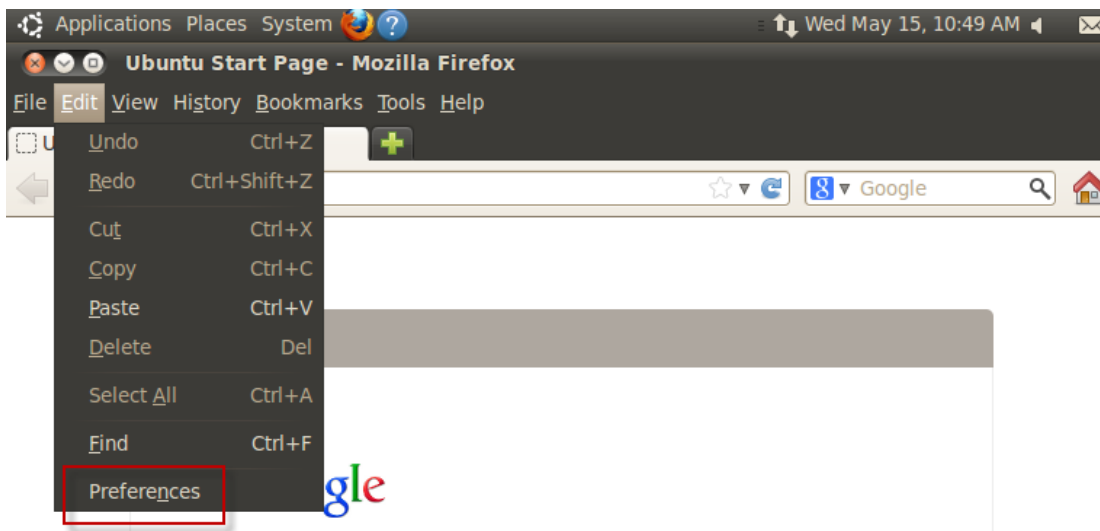


Finally you will be asked to enter your password in order to set your network proxy information. This is typically the same password you used to log into Ubuntu on boot.



Then “Close”.

You should now be able to browse the Internet using Firefox within Ubuntu. Note: If you are still unable to browse the internet, open Firefox. Go to, ‘Edit’, ‘Preferences’, under the ‘Advanced’ tab, click ‘Network’, then click ‘Settings’. Make sure the proxy address is copied into the relative field, use ‘Automatic proxy configuration URL’ or ‘Use system proxy settings’.



1.2.4 Configure Git Proxy in Ubuntu

Configure apt-get by adding following to /etc/apt/apt.conf (create the file if does not exist). Root access required to modify/create this file. A simple text editor, "gedit", can be used to perform these changes:

```
sudo gedit /etc/apt/apt.conf
```

Add following part to make the apt-get can get Ubuntu components through proxy (red part should be your proxy server and port):

```
ACQUIRE
{
    http::proxy "http://wwwgate.ti.com:80"
}
```

Open and edit the \$HOME/.bashrc file (create it if does not exist), then add the following items in this file to setup proxy:

```
export http_proxy="http://webproxy.ext.ti.com:80"
export ftp_proxy="http://webproxy.ext.ti.com:80"
export https_proxy="http://webproxy.ext.ti.com:80"
export no_proxy="ti.com"
export GIT_PROXY_COMMAND=$HOME/git-proxy.sh
```

Source .bashrc file after updating it:

```
source ~/.bashrc
```

Add following in \$HOME/git-proxy.sh (create the file if it does not exist) and save the file (red part should be your proxy server and port):

```
#!/bin/sh
if [ $(getent hosts intranet.ti.com|cut -d' ' -f 1)x = "127.0.0.1x" ]
then
    # this machine is inside intranet network
    if echo $1 | grep ti.com > /dev/null
    then
        # ... and so is the remote machine
        socat - tcp:$1:$2
    else
        socat - proxy:wwwgate.ti.com:$1:$2,proxyport=80
    fi
else
    socat - tcp:$1:$2
fi
```

2. Git Server and Client Setup

To make the git operations work correctly, there are several steps need to be configured on both git server side and client side.

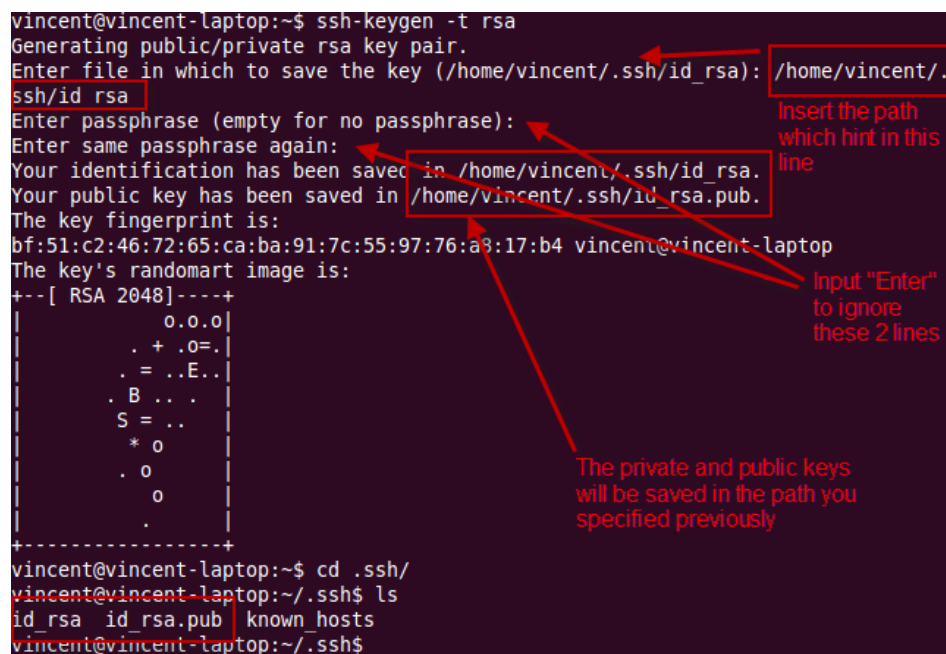
Some basic info is as follow:

1. We use the Chengdu Linux server (10.85.172.62) as the **git server**
2. We use the Ubuntu 10.04 virtual machine installed on Windows PC as the **git client**
3. To simplify the interaction between git client and server, we need to use the SSH authentication to avoid input git server password every time

2.1 Create Private & Public Key on Git Client

Generate SSH private & public key on client by using:

```
ssh-keygen -t rsa
```



```
vincent@vincent-laptop:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/vincent/.ssh/id_rsa): /home/vincent/.ssh/id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/vincent/.ssh/id_rsa.
Your public key has been saved in /home/vincent/.ssh/id_rsa.pub.
The key fingerprint is:
bf:51:c2:46:72:65:ca:ba:91:7c:55:97:76:a3:17:b4 vincent@vincent-laptop
The key's randomart image is:
+--[ RSA 2048 ]-----+
|          . o . o |
|       . + . o = . |
|      . = .. E . . |
|     . B . . .    |
|    S = ..        |
|   * 0            |
|  . 0             |
|   0              |
|  .               |
+-----+
vincent@vincent-laptop:~$ cd .ssh/
vincent@vincent-laptop:~/.ssh$ ls
id_rsa id_rsa.pub known_hosts
vincent@vincent-laptop:~/.ssh$
```

Annotations in the image:

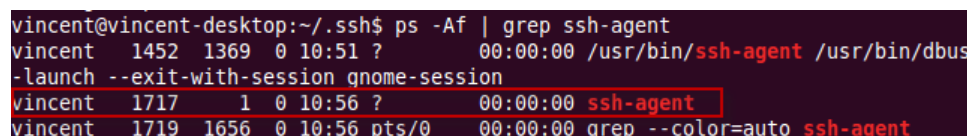
- Red box around `/home/vincent/.ssh/id_rsa`: "Insert the path which hint in this line"
- Red box around `id_rsa id_rsa.pub`: "The private and public keys will be saved in the path you specified previously"
- Red arrow pointing to the empty passphrase line: "Input 'Enter' to ignore these 2 lines"

2.2 Copy and Configure the Public Key to Git Server

We already enable the ssh-agent service on git server, so please make sure the ssh-agent service is also enabled on the git client before any further operations by using:

```
ps -Af | grep ssh-agent
```

If you see the following message on your screen, it means the ssh-agent service is opened:



```
vincent@vincent-desktop:~/.ssh$ ps -Af | grep ssh-agent
vincent  1452  1369  0 10:51 ?        00:00:00 /usr/bin/ssh-agent /usr/bin/dbus
-launch --exit-with-session gnome-session
vincent  1717      1  0 10:56 ?        00:00:00 ssh-agent
vincent  1719  1656  0 10:56 pts/0    00:00:00 grep --color=auto ssh-agent
```

Red box around `ssh-agent` in the third line.

branch: This is branch of the git trunk

HEAD: This is for the latest status

tag: This is the mark for some meaningful status

SHA1: This is the exclusive number for every submit in SHA1

Git has 4 kinds of objects: blob, tree, commit, and tag.

blob represents files, tree represents catalogue, commit represents history, tag represents tags. These 4 objects are all marked by exclusive SHA1 serials. All the information is maintained and contained in the “.git” path of the git folders.

3.2 Install Git Core

Ubuntu has the git core installed by default, please use the following command to check whether the git core is installed or not on your Ubuntu:

```
sudo apt-get install git-core
```

If the following message shows, which means the latest git components are installed:

```
[sudo] password for cifaie:
Reading package lists... Done
Building dependency tree
Reading state information... Done
git-core is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 56 not upgraded.
cifaie@cifaie-cdlinux:/var/git/cifaie KII STK.git$
```

3.3 Init Git Folder

After you check the git core is installed on your Ubuntu, you can create a git folder which will use as the local mirror of the git server, e.g. you can basically use:

```
mkdir ~/git_local
```

```
cd git_local
```

Optionally, you can init this git folder by using:

```
git init
```

You can see there is one more folder named “.git” in this path which contains the git version control info.

```
vincent@vincent-laptop:~/git_local$ git init
Initialized empty Git repository in /home/vincent/git_local/.git/
vincent@vincent-laptop:~/git_local$ ll
total 12
drwxr-xr-x  3 vincent vincent 4096 2013-05-14 16:33 ./
drwxr-xr-x 44 vincent vincent 4096 2013-05-14 16:29 ../
drwxr-xr-x  7 vincent vincent 4096 2013-05-14 16:33 .git/
vincent@vincent-laptop:~/git_local$
```


The “git init” operation is optional because the git related info also can be created when you using the “git clone” command.

3.4 Clone the Project from Git Server

We use the command “git clone” to get the project from the git server:

```
git clone "git project directory"
```

E.g. for the Keystone II STK project, you can use:

```
git clone ssh://cifae@10.85.172.62/var/git/cifae_KII_STK.git
```

```
vincent@vincent-laptop:~/git_local$ git clone ssh://cifae@10.85.172.62/var/git/cifae_KII_STK.git
Initialized empty Git repository in /home/vincent/git_local/cifae_KII_STK/.git/
remote: Counting objects: 7, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 7 (delta 2), reused 0 (delta 0)
Receiving objects: 100% (7/7), done.
Resolving deltas: 100% (2/2), done.
```

There will be a new folder named “cifae_KII_STK” which contains git version control info, enter this folder to check it:

```
cd cifae_KII_STK
```

```
ll
```

```
vincent@vincent-laptop:~/git_local$ cd cifae_KII_STK/
vincent@vincent-laptop:~/git_local/cifae_KII_STK$ ll
total 84
drwxr-xr-x 21 vincent vincent 4096 2013-05-14 16:49 ./
drwxr-xr-x  4 vincent vincent 4096 2013-05-14 16:49 ../
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 AIF2/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 ARM_CorePac/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 BCP_LTE/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 BCP_UMTS/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 C66_CorePac/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 common/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 Ethernet/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 FFTC/
drwxr-xr-x  8 vincent vincent 4096 2013-05-14 16:49 .git/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 Hyperlink/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 IQNET/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 memory_test/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 Navigator/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 PCI-E/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 SPI/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 SRI0/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 TCP3/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 UART/
drwxr-xr-x  2 vincent vincent 4096 2013-05-14 16:49 VCP2/
vincent@vincent-laptop:~/git_local/cifae_KII_STK$
```

This is the basic KII STK code package structure which contains every part we will cover in STK.

You can also use the “-b” option to clone the specified branch, such as:

```
git clone "git project directory" -b "branch name"
```

If you don't use the "-b" option, it will clone the latest version by default.

3.5 Init Git User Info

You can use the following command to edit your user info which can get the correct editor info when checking the commit/update history or log:

```
git config user.name "your name"
```

```
git config user.email "yourname@email_server"
```

```
git config core.editor vim
```

```
git config color.diff true
```

...

Use the following command to check the current info which has been configured:

```
git var -l
```

```
vincent@vincent-laptop:~/git_local/cifae_KII_STK$ git var -l
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
core.editor=vim
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
remote.origin.url=ssh://cifae@10.85.172.62/var/git/cifae_KII_STK.git
branch.master.remote=origin
branch.master.merge=refs/heads/master
user.name=Vincent Han
user.email=vincent-han@ti.com
color.diff=true
GIT_COMMITTER_IDENT=Vincent Han <vincent-han@ti.com> 1368521973 +0800
GIT_AUTHOR_IDENT=Vincent Han <vincent-han@ti.com> 1368521973 +0800
GIT_EDITOR=vim
GIT_PAGER=pager
vincent@vincent-laptop:~/git_local/cifae_KII_STK$
```

3.6 Git File Modification

After clone the code from the git server, you may have your own development and codify the code or structure, you may need to use the following commands:

3.6.1 Add Git Files

Add the relative file which you want to change on the git server by using:

```
git add "directory & file name"
```

Use `git add .` to add the content in current directory, please note there is a '.' here stands for current directory.

Use `git add -a` to add all files in git directory except for which in “.gitignore”, please careful when using this command.

3.6.2 Delete Git Files

Delete the relative file which you want to change on the git server by using:

```
git rm "directory & file name"
```

Use `git rm -r "folder name"` to delete the folder

3.6.3 Check Local Modified Files

Check the local modified file by using:

```
git ls-files -m
```

Use `git ls-files` to see all files which managed by this git project

3.7 Submit Update to Git Server

If you think this need to be update to the git server, you may need the following commands:

3.7.1 Commit Local Modifications

Commit local modification by using:

```
git commit -m "comments"
```

The “-m” options will add your comment in the command line for this commit.

After your commit, there will print the relative modification info:

```
vincent@vincent-laptop:~/git_local/cifae_KII_STK$ git commit -m "update"
# On branch master
nothing to commit (working directory clean)
vincent@vincent-laptop:~/git_local/cifae_KII_STK$
```

3.7.2 Push to Git Server

This step will push the modifications on git server and take effect:

```
git push "git server" "branch name"
```

Due to the default remote name of git server is “origin” and current only have the “master” branch, you can just use `git push origin master` to push modification to git server.

3.8 Get Update from Git Server

This step will get the modifications from git server and take effect in local git folder:

```
git pull "git server" "branch name"
```

Due to the default remote name of git server is “origin” and current only have the “master” branch, you can just use `git pull origin master` to get modifications from git server.

3.9 Branch and Tags

Branch and tag are some meaningful marked points or distributions which can be distinguished by git user.

3.9.1 Git Branch

It's a good habit to use branch to maintain the git client and server code parts which edit by different persons.

To create a new branch:

```
git branch "branch name"
```

To delete a branch:

```
git branch -d "branch name"
```

To check all branches:

```
git branch -a
```

To switch to another branch:

```
git checkout "branch name"
```

Add “-m” option in git check out can force the branch switch without commit:

```
git checkout -f "branch name"
```

3.9.2 Git Tag

Tag can set some points which you need to be recorded or marked

Add tag by using:

```
git tag "tag name" "SHA1 in one commit"
```

Show tag by using:

```
git tag
```

Push tag to git server by using:

```
git push "tag name"
```

3.10 Version Differences Comparison

We can use the git diff to check the file differences and version differences

Compare differences between HEAD and specified tag by using:

```
git diff "tag name"
```

Compare file differences between HEAD and specified tag by using:

```
git diff "tag name" "file name"
```

Compare file differences between 2 tags by using:

```
git diff "tag name 1":"file name 1" "tag name 2":"file name 2"
```

Compare 2 tags differences by using:

```
git diff "tag name 1".."tag name 2"
```

Compare 2 commit differences by using:

```
git diff "SHA1 1".."SHA1 2"
```

3.11 Version Conflict and Merge

Git merge is used to merge different branches or combine with master.

Merge current branch with specified branch by using:

```
git merge "another branch name"
```

Merge current branch with master by using:

```
git merge master
```

Please note that if there have conflicts when merging 2 branches, you need to solve these conflicts manually then commit by using:

```
git update-index
```

```
git commit -a -e /* submit all modification and use VI to edit submit log */
```

3.12 Log and History

Use the git log to track and show the history of files, tags and branches

3.12.1 Git Log

To check the file log:

```
git log "file name"
```

To check the all the log:

```
git log -p
```

To check the log in 2 tags:

```
git log "tag name 1".. "tag name 2"
```

To check the log in HEAD and specified tag:

```
git log "tag name 1"..
```

To check the file log in 2 tags:

```
git log -p "tag name 1".. "tag name 2" "file name"
```

3.12.2 Git Reset

Use git reset to recall the commit before push to the git server

To recall the last time commit:

```
git reset HEAD^
```

To recall the last time commit and clear local modification:

```
git reset --hard HEAD^
```

To reset the SHA1 serial back to the last time commit:

```
git reset SHA1
```

Appendix A. Basic Ubuntu Linux Operation Commands

3.1 Basic

1. In Linux, the "." represents the current directory, the ".." represents the 1 upper level directory.
2. In Linux, the "~" represents the user's home folder directory such as /home/"user name", it is the same with the system environment variables macro "HOME"
3. In Linux, the macro usually use the capital characters, you can directly use it by adding "\$" before the macro such as "\$PATH"
4. In Linux, the file/directory name can be distinguished by capital, which means e.g. "read.txt" and "Read.txt" are both valid in the same directory.
5. In Linux, if you don't know how to use some command, it's usual to use "man" command before this command to show how to use this command, e.g. `man chmod`

6. In Linux, sudo command is used to add before any other commands which can let you have the highest permission to do the operations, if you encountered commands returned “permission denied”, you can use the sudo command to redo it, e.g. `sudo cp a.txt test/`

3.2 Directory Operations

1. pwd

```
pwd
```

Show current directory

2. mkdir

```
mkdir 'directory and folder name'
```

Make new directory

3. cd

```
cd 'directory and folder name'
```

```
cd ..
```

Enter the directory

3.3 File Operations

3.3.1 Create & Delete

1. touch

```
touch "directory and file name"
```

Create a new file

2. rm

```
rm "directory and file name"
```

```
rm -r "directory name"
```

Delete directory or file

3.3.2 Copy & Move

1. cp

```
cp "directory 1 and file name 1" "directory 2 and file name 2"
```

Copy files between 2 directories, use “-r” option when you do the directory operations

2. mv

```
mv "directory 1 and file name 1" "directory 2 and file name 2"
```

```
mv "file name 1" "new file name 1"
```

```
mv -r "directory name 1" "new directory name 1"
```

Move or rename the files & directories

3.3.3 Modify & Edit

1. cat

```
cat "directory and file name 1" "directory and file name 2"
```

Combine 2 files together

2. vi

```
vi "directory and file name"
```

Use VI to edit the file, use `vi --help` for more operations in VI

3. gedit

```
gedit "directory and file name"
```

gedit is the editor with UI, it's much easier to use than VI, you can use gedit to edit file on your Ubuntu virtual machine.

If your Ubuntu don't have the gedit, use the following command to install the gedit:

```
sudo apt-get install gedit
```

4. ghex

```
ghex "directory and file name"
```

ghex is the hex editor with UI, it's much easier to use than VI, you can use ghex to edit hex file such as files extend by ".bin" on your Ubuntu virtual machine.

If your Ubuntu don't have the ghex, use the following command to install the ghex:

```
sudo apt-get install ghex
```

3.4 Permissions

1. sudo

```
sudo "other command"
```


In Linux, sudo command is used to add before any other commands which can let you have the highest permission to do the operations, if you encountered commands returned “permission denied”, you can use the sudo command to redo it, e.g. `sudo cp a.txt test/`

2. chown

```
chown "user":"group" "directory or file name"
```

chown changes files or directories ownership properties, it will be used when you want to change these properties, e.g. `chown user:root /var/a.txt`

3. chmod

chmod changes files or directories permission properties, it will be used when you want to change these properties, e.g. `chmod 775 /var/a.txt` `chmod g+w-x /test/b.dat`

The permission property was coded in octal, you can set permissions for user, group, and world, and you can set whether each can read, write, and/or execute the file. For each permission, read(r)=4, write(w)=2, execute(x)=1; for all the highest permissions encoded together is 777 (`rw-rw-rw-`), which means `rw-` for user, group and world; e.g. 755 (`rw-r--r--`) means read, write and execute by user, read and execute by group and world

For example, if a file had permission to allow everybody to read but only the user could write, the permissions would read `rw-r--r--`. To add or remove a permission, you append a + or a - in front of the specific permission. For example, to add the capability for the group to edit in the previous example, you could type `chmod g+x "file name"`

3.5 List and Property

1. ls

```
ls
```

List file in this directory

```
ls -al
```

List file in this directory with detailed properties

3.6 Compress and Uncompress

1. tar

```
tar "options" "file names"
```

Compress or uncompress the files, e.g.:

```
tar cf "file name".tar "file names" /* create tar file */
```

```
tar xf "file name".tar /* extract tar file */
```

```
tar czf "file name".tar.gz "file names" /* create tar.gz file in Gzip*/
```

```
tar xzf "file name".tar.gz /* extract tar.gz file in Gzip*/
```

2. gzip

```
gzip "options" "file names"
```

Compress or uncompress the files by using gzip, e.g.:

```
gzip "file names" /* create gz file */
```

```
gzip -d "file name".gz /* extract gz file */
```

3.7 Network Operations

1. ifconfig

```
ifconfig
```

Check the basic network info

```
Ifconfig -all
```

Check the detailed network info

2. ping

```
ping 127.0.0.1
```

Ping the host and output result, unlike the ping in windows, the ping in Linux doesn't stop, use "CTRL + Z" to force terminates the process.

3.8 Bash Variables

1. env

```
env
```

Show environment variables

2. echo

```
echo $"name"
```

Output of value of the variable, e.g. `echo $PATH`.