Downloading and installing packages

1. Cannot run apt source by default

The apt source after installing Ubuntu 16.04 will be failed.

```
apt source linux
# Reading package lists... Done
# E: You must put some 'source' URIs in your sources.list
```

This is because deb-src is comment out in /etc/apt/sources.list.

```
cat /etc/apt/sources.list

# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.
deb http://jp.archive.ubuntu.com/ubuntu/ xenial main restricted
# deb-src http://jp.archive.ubuntu.com/ubuntu/ xenial main restricted

# Major bug fix updates produced after the final release of the
# distribution.
deb http://jp.archive.ubuntu.com/ubuntu/ xenial-updates main restricted
# deb-src http://jp.archive.ubuntu.com/ubuntu/ xenial-updates main restricted
```

2. Enable deb-src

Add deb-src URL to repository list.

```
sudo su -c "grep '^deb ' /etc/apt/sources.list | \
sed 's/^deb/deb-src/g' > /etc/apt/sources.list.d/deb-src.list"
```

Update repository database.

```
sudo apt update -y
```

3. Execution result

Now the apt source is succeed

```
mkdir linux
cd linux
```

```
apt source linux

# Reading package lists...
# NOTICE: 'linux' packaging is maintained in the 'Git' version control system at:
# git://git.launchpad.net/~ubuntu-kernel/ubuntu/+source/linux/+git/xenial
# Please use:
# git clone git://git.launchpad.net/~ubuntu-
kernel/ubuntu/+source/linux/+git/xenial
# to retrieve the latest (possibly unreleased) updates to the package.
# Need to get 144 MB of source archives.
# Get:1 http://jp.archive.ubuntu.com/ubuntu xenial-updates/main linux 4.4.0-28.47
(dsc) [9,611 B]
# Get:2 http://jp.archive.ubuntu.com/ubuntu xenial-updates/main linux 4.4.0-28.47
(tar) [133 MB]
# Get:3 http://jp.archive.ubuntu.com/ubuntu xenial-updates/main linux 4.4.0-28.47
(diff) [11.3 MB]
# gpgv: Signature made 2016年06月24日 19時02分30秒 JST using RSA key ID FDCE24FC
# gpgv: Can't check signature: public key not found
# dpkg-source: warning: failed to verify signature on ./linux_4.4.0-28.47.dsc
# dpkg-source: info: extracting linux in linux-4.4.0
# dpkg-source: info: unpacking linux_4.4.0.orig.tar.gz
# dpkg-source: info: applying linux_4.4.0-28.47.diff.gz
# dpkg-source: info: upstream files that have been modified:

ls
# linux-4.4.0  linux_4.4.0-28.47.diff.gz  linux_4.4.0-28.47.dsc
linux_4.4.0.orig.tar.gz
```

Download packages with dependencies locally in Ubuntu

We can do this in two methods. I tested this guide on Ubuntu 16.04 and 18.04 LTS desktop editions. It worked just fine as described below.

Method 1

This is the simplest and straight-forward method than other other methods given below.

To download a package with all dependencies, without installing them, just run:

```
sudo apt-get install --download-only <package_name>
```

For instance, let us download the Vim package with all required dependencies, without installing them, using command:

```
sudo apt-get install --download-only vim
Sample output:

# Reading package lists... Done
# Building dependency tree
```

```
# Reading state information... Done
# Suggested packages:
# ctags vim-doc vim-scripts
# The following NEW packages will be installed:
# vim
# 0 upgraded, 1 newly installed, 0 to remove and 82 not upgraded.
# Need to get 1,152 kB of archives.
# After this operation, 2,852 kB of additional disk space will be used.
# Get:1 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 vim amd64
2:8.0.1453-1ubuntu1.1 [1,152 kB]
# Fetched 1,152 kB in 3s (372 kB/s)
# Download complete and in download only mode
```

As you see in the above output, we have downloaded Vim package will all dependencies, but we didn't actually install it.

- Download Packages With Dependencies Locally In Ubuntu and derivatives
- Download Packages With Dependencies Locally In Ubuntu
- All downloaded files will be saved in /var/cache/apt/archives directory.

Just copy the entire cache folder on any USB or transfer them via network to a system that you wanted to install the packages in it.

To install the downloaded packages, go to the cache folder and install them as shown below.

```
sudo dpkg -i *
```

See? It's that simple!

However, this method only works if the system that you use to download the packages does not have the main package or its dependencies installed locally.

If you try to download a package which is already installed in the same system itself, you will see an output like below.

```
sudo apt-get install --download-only vim

# Reading package lists... Done
# Building dependency tree
# Reading state information... Done
# vim is already the newest version (2:8.0.1453-1ubuntu1.3).
# 0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.

In such cases, use `apt-rdepends` to download all packages. If apt-rdepends is not
installed yet, install it using command:

```sh
sudo apt install apt-rdepends
```
```

And then download the main package (i.e. Vim in our case) along with all dependencies using command:

```
apt download $(apt-rdepends vim | grep -v "^ ")
```

This command will recursively download all required packages.

Just in case if you encountered with an error like below:

```
E: Can't select candidate version from package debconf-2.0 as it has no candidate
```

Try this command instead:

```
apt-get download $(apt-rdepends vim | grep -v "^ " | sed 's/debconf-
2.0/debconf/g')
```

This command will download Vim with all needed packages and save them in the current working directory.

To install all downloaded packages, run:

```
sudo dpkg -i *
```

Method 2:

First, download the dependencies of the package you wanted to download.

To display list of all dependencies of a package, for example Python, run:

```
sudo apt-cache depends python

# Sample output:

# Python
# PreDepends: python-minimal
# Depends: python2.7
# Depends: libpython-stdlib
# Conflicts: <python-central>
# Breaks: update-manager-core
# Suggests: python-doc
# Suggests: python-tk
# Replaces: python-dev

Let us download python package with its dependencies to our local disk.

To do so, first create a directory to save the packages.
```

```sh
mkdir python
```

Go to the directory:

```
cd python
```

And then run:

```
$ for i in $(apt-cache depends python | grep -E 'Depends|Recommends|Suggests' |
cut -d ':' -f 2,3 | sed -e s/'<'/''/ -e s/'>'/''/); do sudo apt-get download $i
2>>errors.txt; done
```

The above command will download Python package along with all required dependencies and saves them in the current working directory. This command will also save any errors in the errors.txt file.

Let us view the downloaded files using 'ls' command:

```
ls

Sample output:

# errors.txt
# libpython-stdlib_2.7.11-1_amd64.deb
# python2.7_2.7.11-7ubuntu1_amd64.deb
# python-doc_2.7.11-1_all.deb
# python-minimal_2.7.11-1_amd64.deb
# python-tk_2.7.11-2_amd64.deb
```

List downloaded packages

As you see in the above output, python package with all its dependencies has been downloaded.

Just copy them to your USB drive and install the python packages on any offline system as shown below.

Mount the USB drive, go to the location where you have mounted the drive, and run the following command to install Python.

```
sudo dpkg -i *
```

Suggested Read

- Download packages with dependencies locally for a specific architecture

- You might notice that the above command has downloaded the 64 bit packages. It is because I am downloading them from the 64-bit Ubuntu system. What if you want to download packages for 32-bit arch systems? It's also possible!

First, enable the architecture you want in your Ubuntu system using command:

```
sudo dpkg --add-architecture i386
```

If you don't add the architecture, you will get the following error message when try to download the packages.

```
E: No packages found
```

After enabling the Architecture of your choice, run the following command to download specific architecture related packages.

```
$ for i in $(apt-cache depends python:i386 | grep -E 'Depends|Recommends|Suggests'
| cut -d ':' -f 2,3 | sed -e s/'<'/''/ -e s/'>'/''/); do sudo apt-get download $i
2>>errors.txt; done
```

As you see in the above output, I have added the architecture 'i386' with 'apt-cache' command.

Sample output:

```
Get:1 http://in.archive.ubuntu.com/ubuntu xenial/main i386 python-minimal i386
2.7.11-1 [28.2 kB]
Fetched 28.2 kB in 1s (25.8 kB/s)
Get:1 http://in.archive.ubuntu.com/ubuntu xenial/main i386 python2.7 i386 2.7.11-
7ubuntu1 [220 kB]
Fetched 220 kB in 1s (116 kB/s)
Get:1 http://in.archive.ubuntu.com/ubuntu xenial/main i386 libpython-stdlib i386
2.7.11-1 [7,664 B]
Fetched 7,664 B in 0s (13.3 kB/s)
Get:1 http://in.archive.ubuntu.com/ubuntu xenial/main i386 python-tk i386 2.7.11-2
[28.0 kB]
Fetched 28.0 kB in 1s (24.8 kB/s)
```

Let us check the downloaded packages.

```
ls

Sample output:
```

```
errors.txt
libpython-stdlib_2.7.11-1_i386.deb
python2.7_2.7.11-7ubuntu1_i386.deb
python-minimal_2.7.11-1_i386.deb
python-tk_2.7.11-2_i386.deb
```

See? The above command downloaded the 32 bit packages only.

You know now how to download packages with dependencies in Ubuntu systems. These methods are same for all DEB-based systems