back to posts

APT Cheat Sheet

Mar 30, 2015 ▪ packagecloud

debian-howto

Table of Contents

dpkg (low level package manager for Debian)

Getting Started:

What does APT do?

APT is a set of core tools found inside the Debian operating system. It provides utilities for the installation and removal of software packages and dependencies on a system.

apt-get / high level package handling utility

`apt-get` is the command-line tool for handling packages and provides functions such as installing, removing, and updating packages on a system with a single operation. We'll cover the following commands for `apt-get`:

- `install` and `--reinstall`
- `remove`

- `purge` or `--purge`
- `upgrade`
- `update`
- `clean` and `autoclean`

apt-cache / high level package query utlity

`apt-cache` provides an interface to perform read-only operations on the APT package cache. `apt-cache` does not change the state of the system, but allows the user to extract useful information from package metadata.

We'll go over the following commands for `apt-cache`:

- `pkgnames`
- `search` `show`

dpkg / low level package manager for Debian

`dpkg` is a tool for installing, removing, and querying individual packages. We'll investigate some common commands and go over some basic usage of `dpkg` in a couple of real-world examples.

- `--list` or `-l`
- `--install`
- `--remove`
- `--purge`
- `--update`
- `--contents`

`apt-get` high level package handling utility

Installing a Debian package:

```
$ sudo apt-get install [package-name]
```

`apt-get` wants you to pass the `[package-name]` you wish to install, for example:

```
$ sudo apt-get install vim
```

Removing a Debian package:

The following will remove a package **without** removing its configuration files:

```
$ sudo apt-get remove [package-name]
```

To remove a package **and** its configuration files, use `purge`:

```
$ sudo apt-get purge [package-name]
```

or alternatively, use the `--purge` flag on the remove command:

```
$ sudo apt-get --purge remove [package-name]
```

Update package index files from sources.list:

```
$ sudo apt-get update
```

When this command is run, all available packages are fetched and re-indexed from the locations specified in `/etc/apt/sources.list` and `/etc/apt/sources.list.d/`.

Upgrade all debian system packages:

```
$ sudo apt-get upgrade
```

This command installs **all** of the latest versions of each package installed on the system and is, generally, not recommended to be run on production systems.

Update / Reinstall a single package:

Once you've run `apt-get update` to update repository metadata, you can update an installed package by running `apt-get install`

```
$ sudo apt-get install [package-name]
```

If you need to force reinstall a package, just pass the `--reinstall` flag

```
$ sudo apt-get --reinstall install [package-name]
```

By passing the `--reinstall` flag, you will effectively force the package to be reinstalled even if it's already installed and at the latest version. This will completely remove the package from the

system* and reinstall it.

*Packages that depend on the `[package-name]` being reinstalled will not be removed from the system

APT cache files:

APTs cached files are located in:

- `/var/cache/apt/archives/`

Clear the APT cache:

```
$ sudo apt-get clean
```

The `clean` command clears out the local repository of downloaded package files. It removes everything **except** the *partials folder* and *lock file* from `/var/cache/apt/archives/`.

Use `apt-get clean` to free up disk space when necessary, or as part of regularly scheduled maintenance.

Remove useless files from the APT cache:

```
$ sudo apt-get autoclean
```

`autoclean` is another method used to clear out the local repository of downloaded package files, just like `clean`. The difference between `clean` and `autoclean` is that the latter only removes package files that can no longer be downloaded from their sources, and are very likely to be useless.

apt-cache high level package query utility

List all available packages:

```
$ apt-cache pkgnames
```

This command will output a list of available package names for your system:

```
...
aspell-bg
389-ds-console-doc
libreoffice-l10n-ga
libindicate-doc
libreadline-dev
libpng12-dev
```

Searching for a specific debian package:

```
$ apt-cache search [package-name-pattern]
```

This is really useful in case you don't know the exact `[package-name]`, but rather a description of what that package does; for example "Network Security":

```
$ apt-cache search "Network Security"
```

This will return a list of packages containing the string "Network Security" in the package description. Using apt-cache will look in the `name`, `description`, and `provides` fields of the available packages by default.

```
libnss3 - Network Security Service libraries
libnss3-1d - Network Security Service libraries
libnss3-dbg - Debugging symbols for the Network Security Service libraries
libnss3-dev - Development files for the Network Security Service libraries
coolkey - Smart Card PKCS #11 cryptographic module
daemonlogger - simple network packet logger and soft tap daemon
grepcidr - Filter IP addresses matching IPv4 CIDR/network specification
hlbrw - assistant to help make new rules to HLBR
libjss-java - Network Security Services for Java
libnss3-tools - Network Security Service tools
libopenvas2 - remote network security auditor - shared libraries
libopenvas2-dev - remote network security auditor - static libraries and headers
openvas-client - Remote network security auditor, the client
openvas-plugins-base - remote network security auditor - basic plugins
openvas-plugins-dfsg - remote network security auditor - plugins
openvas-server - remote network security auditor - server
openvas-server-dev - remote network security auditor - static libraries and headers
python-nss - Python bindings for Network Security Services (NSS)
scute - OpenPGP smartcard plugin for Mozilla Network Security Services
```

Show debian package information:

```
$ apt-cache show [package-name]
```

This will show apt metadata for the `[package-name]` given. This is an example using the “screen” package:

```
$ apt-cache show screen
```

will output:

```
Package: screen
Priority: optional
Section: misc
Installed-Size: 1052
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Jan Christoph Nordholz <hesso@pool.math.tu-berlin.de>
Architecture: amd64
Version: 4.0.3-14ubuntu8
Depends: libc6 (>= 2.4), libncursesw5 (>= 5.6+20070908), libpam0g (>= 0.99.7.1), dpkg (>= 1.15.
4) | install-info, upstart-job
Suggests: byobu
Filename: pool/main/s/screen/screen_4.0.3-14ubuntu8_amd64.deb
Size: 611204
MD5sum: b5e98bb56fdfc9bf9fd13e6f726c83aa
SHA1: 8d3e5c5d858b4a314a66b5bc51b3a557b85ea96c
SHA256: 2b6c752fc226ad6e2e32cd93f089bf2a89d51e95f6e5ff1e7ed63b0b57ff592f
Description-en: terminal multiplexor with VT100/ANSI terminal emulation
 screen is a terminal multiplexor that runs several separate "screens" on a
 single physical character-based terminal.  Each virtual terminal emulates a
```

```
 DEC VT100 plus several ANSI X3.64 and ISO 2022 functions.  Screen sessions
 can be detached and resumed later on a different terminal.
 .
 Screen also supports a whole slew of other features.  Some of these are:
 configurable input and output translation, serial port support, configurable
 logging, multi-user support, and utf8 charset support.
Homepage: http://savannah.gnu.org/projects/screen
Description-md5: 031a852784c43a4c757fecf6b610c93e
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Origin: Ubuntu
Supported: 5y
Task: cloud-image, ubuntu-usb, server, edubuntu-usb, edubuntu-desktop-kde, edubuntu-desktop-gno
me
```

NOTE: Installed-Size and Size are returned in bytes - link to discussion

dpkg low level package manager for Debian

Install a package

```
dpkg -i [/path/to/vim_7.3.429-2ubuntu2_amd64.deb]
```

or alternatively, use the `--install` flag

```
dpkg --install [/path/to/vim_7.3.429-2ubuntu2_amd64.deb]
```

Remove a package

To remove a package using `dpkg` **without** removing its configuration files:

```
$ dpkg --remove [package-name]
```

alternatively, use the `-r` flag:

```
$ dpkg -r [package-name]
```

To remove a package using `dpkg` **along with** its corresponding configuration files, use the `--purge` command:

```
$ dpkg --purge [package-name]
```

List available system packages:

`dpkg -l` allows you to list a set of packages on the system and the state of those packages:

```
    $ dpkg -l [package-name-pattern]
```

You can use a regular expression to list information about all matching package names. For example:

```
    $ dpkg -l "re*"
```

will return all packages starting with the letters “re”:

```
||/ Name                          Version                      Description
+++-=============================-============================-=====================================
================================
un  readahead                     <none>                       (no description available)
ii  readline-common               6.2-8                        GNU readline and history libraries, c
ommon files
un  rebuildd                      <none>                       (no description available)
un  redhat-cluster-modules        <none>                       (no description available)
ii  redis                         2.8.3-1                      redis
un  regina-normal-dev             <none>                       (no description available)
```

```
ii  reprepro                 4.8.2-1ubuntu0.1          Debian package repository producer
ii  resolvconf               1.63ubuntu15              name server information handler
```

The first column shows the state of the package. You can learn more about package states by reading the dpkg man page: `man 1 dpkg`.

If the `[package-name-pattern]` is omitted from `dpkg -l` then all packages in `/var/lib/dpkg/status` will be listed, excluding packages that have been marked not-installed

```
$ dpkg -l
```

will output something like:

```
...
ii  libxml2                 2.7.8.dfsg-5.1ubuntu4.11  GNOME XML library
ii  libxml2-dbg             2.7.8.dfsg-5.1ubuntu4.11  Debugging symbols for the GNOME XML l
ibrary
ii  libxml2-dev             2.7.8.dfsg-5.1ubuntu4.11  Development files for the GNOME XML l
ibrary
ii  libxmuu1                2:1.1.0-3                 X11 miscellaneous micro-utility libra
ry
ii  libxrender1             1:0.9.6-2ubuntu0.1        X Rendering Extension client library
ii  libxslt1-dev            1.1.26-8ubuntu1.3         XSLT 1.0 processing library - develop
ment kit
ii  libxslt1.1              1.1.26-8ubuntu1.3         XSLT 1.0 processing library - runtime
```

```
library
ii  libyaml-0-2               0.1.4-2                  Fast YAML 1.1 parser and emitter libr
ary
ii  libyaml-dev               0.1.4-2                  Fast YAML 1.1 parser and emitter libr
ary (development)
ii  linux-firmware            1.79                     Firmware for Linux kernel drivers
ii  linux-image-3.2.0-29-virtu 3.2.0-29.46             Linux kernel image for version 3.2.0
on 64 bit x86 Virtual Guests
ii  linux-image-virtual       3.2.0.29.31              Linux kernel image for virtual machin
es
ii  linux-libc-dev            3.2.0-37.58              Linux Kernel Headers for development
...
```

List files in a package:

dpkg maintains a list of packages that are installed on a system in `/var/lib/dpkg`.

You can query the files in an installed package using `dpkg -L`:

```
    $ dpkg -L [package-name]
```

So, for example:

```
    $ dpkg -L redis
```

returns the following results:

```
/.
/usr
/usr/share
/usr/share/doc
/usr/share/doc/redis
/usr/share/doc/redis/copyright
/usr/share/doc/redis/changelog.Debian.gz
/usr/bin
/usr/bin/redis-check-dump
/usr/bin/redis-benchmark
/usr/bin/redis-server
/usr/bin/redis-check-aof
/usr/bin/redis-cli
```

If you'd like to list the files in a debian package that you've downloaded (but not installed), you can use the `--contents` flag.

For example:

```
    $ dpkg --contents /path/to/redis_2.8.3-1_amd64.deb
```

returns the following results:

```
drwxr-xr-x root/root         0 2014-07-24 09:52 ./
drwxr-xr-x root/root         0 2014-07-24 09:52 ./usr/
drwxr-xr-x root/root         0 2014-07-24 09:52 ./usr/share/
drwxr-xr-x root/root         0 2014-07-24 09:52 ./usr/share/doc/
drwxr-xr-x root/root         0 2014-07-24 09:52 ./usr/share/doc/redis/
-rw-r--r-- root/root         0 2014-07-24 09:52 ./usr/share/doc/redis/copyright
-rw-r--r-- root/root       139 2014-07-24 09:52 ./usr/share/doc/redis/changelog.Debian.gz
drwxr-xr-x root/root         0 2014-07-24 09:52 ./usr/bin/
-rwxr-xr-x root/root     22616 2014-07-24 09:52 ./usr/bin/redis-check-dump
-rwxr-xr-x root/root    285712 2014-07-24 09:52 ./usr/bin/redis-benchmark
-rwxr-xr-x root/root    844496 2014-07-24 09:52 ./usr/bin/redis-server
-rwxr-xr-x root/root     10320 2014-07-24 09:52 ./usr/bin/redis-check-aof
-rwxr-xr-x root/root    320432 2014-07-24 09:52 ./usr/bin/redis-cli
```

When the `--contents` flag is used, `dpkg` calls down to an action provided by another tool `dpkg-deb`, which provides tools to manipulate a debian package archive.

Show packages containing a filename or filepath:

```
$ dpkg -S [filename-search-pattern]
```

For example, passing a specific filepath:

```
$ dpkg -S /usr/share/man/man5
```

will return all the package names that contain that file path.

```
libc-bin, libarchive-dev, ntp, libpam-modules, libpam-runtime, initscripts, upstart, mount,
libx11-data, fontconfig-config, isc-dhcp-client, isc-dhcp-common, rsyslog, sudo, ucf,
collectd-core, libwrap0, openssl, openssh-client, rsync, module-init-tools, ncurses-bin,
procps, login, nfs-common, cryptsetup, git-man, man-db, cron, libmagic1, libldap-2.4-2,
libtirpc1, libnfsidmap2, adduser, resolvconf, dpkg, e2fsprogs, ifupdown, initramfs-tools,
locales, kbd, openssh-server, apt, wireless-regdb, util-linux, passwd, mime-support, net-tools,
manpages, dpkg-dev: /usr/share/man/man5
```


Show package information:

You can show package metadata of installed packages by using `dpkg -s`:

```
$ dpkg -s [package-name]
```

for example, the following command:

```
$ dpkg -s screen
```

returns the package metadata:

```
Package: screen
Status: install ok installed
Priority: optional
Section: misc
Installed-Size: 1052
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Architecture: amd64
Version: 4.0.3-14ubuntu8
Depends: libc6 (>= 2.4), libncursesw5 (>= 5.6+20070908), libpam0g (>= 0.99.7.1), dpkg (>= 1.15.
4) | install-info, upstart-job
Suggests: byobu
Conffiles:
 /etc/screenrc 12c245238eb8b653625bba27dc81df6a
 /etc/init/screen-cleanup.conf 441f4a1c5b41d7f23427be5aa6ccbbcc
Description: terminal multiplexor with VT100/ANSI terminal emulation
 screen is a terminal multiplexor that runs several separate "screens" on a
 single physical character-based terminal.  Each virtual terminal emulates a
 DEC VT100 plus several ANSI X3.64 and ISO 2022 functions.  Screen sessions
 can be detached and resumed later on a different terminal.
 .
```

```
 Screen also supports a whole slew of other features.  Some of these are:
 configurable input and output translation, serial port support, configurable
 logging, multi-user support, and utf8 charset support.
Homepage: http://savannah.gnu.org/projects/screen
Original-Maintainer: Jan Christoph Nordholz <hesso@pool.math.tu-berlin.de>
```

Conclusion

Getting more familiar with your package manager's tools can help you be more productive when finding, installing, and querying packages.

We highly recommend that users of production Debian and Ubuntu systems become familiar with `apt-get`, `apt-cache`, and `dpkg`. You can learn more about the tools mentioned in this blog post by reading the man pages:

`man 8 apt-get`

`man 8 apt-cache`

`man 1 dpkg`