Apt-Cacher-NG User Manual

Apt-Cacher NG is a caching proxy for software packages which are downloaded by Unix/Linux system distribution mechanisms from mirror servers accessible via HTTP.

This manual provides an overview of Apt-Cacher-NG's features and a walk through the required configuration steps for server administrators and users of the proxy.

Contents

Chapter 1: Introduction 3
Chapter 2: Running apt-cacher-ng 4
Chapter 3: Configuration (Server side) 5
3.1 Configuration file types 5
3.2 Repositories and URL mapping 6
3.3 Taboos 7
Chapter 4: Configuration (Client side) 8
Chapter 5: Security 9
Chapter 6: Maintenance tasks 10
6.1 Local cache cleanup 10
6.2 Import 10
Chapter 7: Troubleshooting & FAQ 11
7.1 Problem: *apt-listbugs* reports errors using HTTP proxy 11
7.2 Problem: *apt-get* freezes when downloading files 11
7.3 *apt-get* reports corrupted bzip2 data 11
Chapter 8: Miscellaneous 13
Chapter 9: Contact 14

Chapter 1: Introduction

apt-cacher-ng attempts to achieve the same goals as related proxies - it acts as a proxy which is used by clients in the local network to share the data that has been downloaded. It monitors the state of packages and is capable of merging downloads of the same packages from different locations (real or simulated).

The package reuses many ideas behind the other famous proxy, its predecessor apt-cacher 1.x (which has been written in Perl). In contrast to apt-cacher, other aspects have been declared as primary targets during the development of apt-cacher-ng:

- lightweight implementation - allow the use on systems with low memory and processing resources
- internal (native) threading - avoiding process fork'ing wherever possible, avoiding kludges for pseudo-thread synchronization, avoiding excessive use of hidden file system features for internal operations
- real (effective) support of HTTP pipelining, therefore a native client with native stream control has been developed. The nice side effect is the reduction of resource overhead and minimization of possible points of failure
- avoiding featurities where they cause too much bloat and the functionality can be provided by native OS features
- reliable but efficient content merging in the local package pool, avoiding delivering of wrong data.

With 0.1, almost all important functionality is implemented.

As with apt-cacher, explicit tracking of dynamically changed and unchanged files is established, and the use in non-Debian environment should be supported.

Long story: Not all goals have been achieved. The initial plan of using background databases to merge any download from any arbitrary location has been dropped because of complexity and performance considerations, reliable heuristics could not be found either. Instead, a semi-automated solution has been created which used machine-parseable files with mirror information, like the one available for Debian mirrors in Debian's CVS repository.

Chapter 2: Running apt-cacher-ng

Run "build/apt-cacher-ng -c conf" when configured where conf is the configuration directory. See section 3.1 for details on possible and required contents of this directory.

Most options from the configuration file can also be passed through command line parameters. Just append them with the same format as in the configuration file but without separating spaces inside, e.g.

```
Port:4855
```

For convenience, the colon can also be replaced with the equals sign and letter case does not matter, so this is also possible:

```
port=4855.
```

Chapter 3: Configuration (Server side)

First, some vocabulary:

- "Repository": the tree in the local cache designated for file storage. May be identical with the hostname of a mirror, but does not need to. In the last case, a backend definition must be provided.
- "Backend": description of a remote side as URL or combination of host and base directory, to look for the files stored under it. Backends are defined as part of remapping scheme, see below.
- "Volatile files": Volatile files provide information about others contained on the particular side, i.e. Packages, Sources and Pdiff index files. "apt-get update" fetches newer copies of them, for example. Volatile files are always object to change.
- "Package files": Those are all other files considered being a software package supported by ACNG. Are not allowed to change on the remote side during the processing, expect trouble otherwise.

Apt-Cacher-Ng can be configured by command line arguments or in its configuration file. Some extra configuration files can be included and automatically used, see next chapters for details.

3.1 Configuration file types

The main configuration files are located in one single directory. Three types files are recognized (by suffix) and interpreted by the following schema.

*.conf

Files matching this pattern contains configuration directives and values in form of "key: value" pairs. See commented example file in the conf/ directory of this package. When multiple files found, the contents are merged.

*.from and other RFC-822 formated files:

Additional information files are supported by ACNG which provide information about remapping rules and backend definitions. The format of the files is self-explaining:

```
Site: <hostname>
Archive-http: /base.directory/
```

(Optional fields are also used in remap descriptions, Alias, Aliases, X-Archive-http:).

HTTP URL lists

The simplest form of input files are plain lists of URLs, one per line. All must begin with http:// and end with a / (slash).

3.2 Repositories and URL mapping

For details about repository mapping, see sample config files shipped with the package. Basic syntax is *one single line* containing:

```
Remap-<repository_name>: /basepath/ file:list.txt ;
 backendhost/dir file:where.txt
```

<repository_name> therein is an arbitrary name chosen by the administrator used to identify the repository in the local cache. It must be clearly different than any possible hostname and must not begin with an underscore (for internal reasons). Examples: *project_foo_mirrors* or *ubuntu_stuff*.

NOTE: unlike with some other APT proxy, *repository_name* does NOT represent any mandatory URL prefix that you need to put into every sources.list. It's only relevant for the internal processing.

The second part of the directive (between : and ; or end of line) consists of path prefixes that needs to be matched against the incoming URL to activate this rule. No wildcards are allowed, the path prefix needs to end with the last directory in the path. When a rewrite rule for a directory is specified here, all files accessed below this directory are stored in the local repository under a location relative to the rewrite path. Therefore special care about identical path level is required when trying to canalize access to multiple download locations into the same cache repository. To avoid cluttering of the configuration file with many URLs, it's possible to specify them in a separate file, for details see Notes below.

Also note that the hostname of the requested URLs is moved to the first path component, therefore it's required to include it in the path prefix spec when remapping access to certain machines. However, no real hostname must be specified when a list of backend servers is configured (see below). This method (similar to techniques used by some other APT proxies) allows to create a shortcut for the users so they don't need to specify (select) a certain Debian mirror.

Examples: */some-actively-used-old-server-name/debian* or *ftp.ubuntu.com/ubuntu/* or */debian/*.

The last part of the Remap-... directive, appended after a semicolon, describes locations to download from. This part is optional if the first part contains only real locations and mandatory if shortcuts are used there.

The format is basically the same as in the first part. The subdirectory name and depth must match the URLs specified there *exactly*. For example, when the rewrite rule specifies "http://ftp.debian.org/debian" and some local mirror offers the data from that URL in "http://192.168.0.17/pub/mirrors/ftp.debian.org/debian" then the rewrite rule might be:

```
Remap-Something: http://ftp.debian.org/debian
   ; http://192.168.0.17/pub/mirrors/ftp.debian.org/debian
```

in *one single line*.

Notes:

- Repository name must be one single word and must be a valid file name. Also see section 3.3.
- The " ; " separates the descriptions of incoming URL maps and backend descriptions

- Location descriptions can be specified as URLs (with or without preceding "http://" or be read from a file (file: must be prepended to the filename in this case). The filename must be relative to the configuration directory if it's not absolute (starting with /). Wildcards in the filename are possible and variable names are expanded (shell-like globing). Compressed files are supported (gzip and bzip2 formats), compression format is detected automatically by the filename suffix (trailing .gz or .bz2). The data format in the file is simple: either one URL per line, or formated as RFC-822 key/value blocks specifying the hostname and path in config blocks (see section 3.1 for details).
- Multiple Remap-FOO lines are possible (sharing the same directive key). In this case their contents are merged.

If no repository is specified, the local storage space is derived from the request URL. If no backend is specified (with or without repositories specification) then the site in the request URL is used to download from, otherwise one of the backends after calculating the new real URL relatively to the backend description.

Use of predefined repositories is recommended. Large remapping lists (for many popular Debian mirrors) are supported quite efficiently; assigning them all into the same repository increases the probability of sharing the data between users that try to use different mirrors. The backends list can be customized to ensure that few locally best reachable mirrors are used to download from.

3.3 Taboos

Some things are discouraged and should or must not be used in local configuration for backwards compatibility or other reasons:

- "apt-cacher" being the name of the proxy server or be referred somehow by the client
- repository names beginning with underscore

Chapter 4: Configuration (Client side)

From the client side, apt-cacher-ng can be used as drop-in replacement for apt-cacher. The same rules apply, e.g. Debian/Ubuntu users should EITHER:

- Specify the caching machine as HTTP Proxy for APT, e.g. putting a line like the following into a file like /etc/apt/apt.conf.d/02proxy:

  ```
  Acquire::http { Proxy "http://CacheServerIp:3142"; };
  ```

OR:

- Replace all mirror hostnames with cachinghost/hostname in sources.list, so

  ```
  deb http://ftp.uni-kl.de/debian etch main
  ```

  now would become:

  ```
  deb http://192.168.0.17/ftp.uni-kl.de/debian etch main
  ```

(assuming that CacheServerIp is 192.168.0.17).

Mixing both configuration methods is not recommended and will lead to obscure APT failures in most cases.

Additionaly, leading path component containing "apt-cacher/" or "apt-cacher?/" might be ignored by the server during the URL processing. This is intended behavior and exists to maintain backwards compatibility to sources.list entries configured for early versions of Apt-Cacher (based on CGI technology).

Chapter 5: Security

Like with many storing daemons with predictable filenames, apt-cacher-ng is vulnerable to symlink attacks and similar malicious actions. Therefore, the user must make sure that the cache and log directories are writable only to the user account under which apt-cacher-ng is executed on.

As to the program internal security, apt-cacher-ng has been developed to care about a certain level of attacks from internal users as well as from malicious outside hosts. However, no guarantees can be made about the security of the program. It's recommended to run apt-cacher-ng under a system account which has no access to any system files outside of the cache and log directories. Refer to the manuals of the administration utilities of your distribution (like start-stop-daemon) to created the required configuration.

Chapter 6: Maintenance tasks

There are few optional tasks that need to be executed by the administrator from time to time or during the initial configuration.

6.1 Local cache cleanup

Visit the report page in a browser and trigger the operation there.

6.2 Import

1. Make sure that apt-cacher-ng has valid index files in the cache. This is the tricky part. To get them right, a client needs to download them through apt-cacher-ng once. Therefore:

   - You need to configure the server and one client before doing the import. See above for instructions.
   - Run "apt-get update" on client(s) once to teach ACNG about remote locations of (volatile) index files

NOTE: APT is pretty efficient on avoiding unneccessary downloads which can make a proxy blind to some relevant files. ACNG makes some attempts to guess the remote locations of missed (not downloaded) files but these heuristics may fail, especially on non-Debian systems. When some import files are permantently ignored, check the process output for messages about the update of Packages/Sources files. When some relevant package sources are missing there, there is a brute-force method to force their download to the client. To do that, run:

```
rm /var/cache/apt/*cache.bin
rm /var/lib/apt/lists/*Packages
rm /var/lib/apt/lists/*Sources
```

on the client to purge APT's internal cache, and rerun "apt-get update".

1. Store copies of your .debs, .orig.tar.gz, ... somewhere in the "_import" subdirectory in the cache, ie. in /var/cache/apt-cacher/_import/. The files may be links or symlinks, does not matter. When done, apt-cacher will move those files to its own internal locations. Example:

   ```
   cd /var/cache
   mkdir apt-cacher-ng/_import
   cp -laf apt-proxy apt-cacher /var/cache/apt-cacher-ng/_import
   chown -R apt-cacher-ng apt-cacher-ng/_import
   ```

2. Visit the report page and trigger the import action there. Check the results, look for (red) error messages. If some files could not be moved around, they are either unusable or could not be removed because of missing permissions. Remove those copies from _import directory manually.

Chapter 7: Troubleshooting & FAQ

7.1 Problem: *apt-listbugs* reports errors using HTTP proxy

Solution: advise them not to use the main proxy for access to bugs.debian.org, adding some bits like following to */etc/apt/apt.conf*:

```
Acquire::HTTP::Proxy::bugs.debian.org "DIRECT";
//or Acquire::HTTP::Proxy::bugs.debian.org  "other.proxy:port"
```

Real configurable pass-through for unsupported actions mode will be added in some future version of Apt-Cacher NG.

Thanks to Jamil Djadala for finding this workaround.

7.2 Problem: *apt-get* freezes when downloading files

Solution: First, check:

- Free disk space and inode usage (*"df"*, *"df -i"*)
- Internet connection to the remote sites (browse them via HTTP, e.g. visiting http://ftp.your.mirror)

If nothing helps then you may have hit a spooky problem which is hard to track down. If you like, help the author on problem identification. To do that, do:

```
su -
# enter root password
cd /tmp
apt-get source apt-cacher-ng
apt-get build-dep apt-cacher-ng
cd apt-cacher-ng-*
make acng-debug
/etc/init.d/apt-cacher-ng stop
./apt-cacher-ng.debug -c /etc/apt-cacher-ng logdir=/tmp
# (let apt-get run now, on timeouts just wait >> 20 seconds)
# stop the daemon with Ctrl-C
/etc/init.d/apt-cacher-ng start
# compress /tmp/apt-cacher.err and send it to author
chown -R apt-cacher-ng:apt-cacher-ng /var/cache/apt-cacher-ng
```

7.3 *apt-get* reports corrupted bzip2 data

Symptoms: apt-get fails to run through "update" no matter what you do. And you may have get a message like this one.

```
99% [6 Packages bzip2 0] [Waiting for headers] [Waiting for headers]
bzip2: Data integrity error when decompressing.
        Input file = (stdin), output file = (stdout)

It is possible that the compressed file(s) have become corrupted.
You can use the -tvv option to test integrity of such files.

You can use the `bzip2recover' program to attempt to recover
data from undamaged sections of corrupted files.

Err http://debian.netcologne.de unstable/main Packages
  Sub-process bzip2 returned an error code (2)
```

Chapter 8: Miscellaneous

Filtering by client IP or hostname is not supported directly. However, an inetd daemon is shipped with the package which makes the use of tcpd possible. See tcpd(8) and related manpages for further details. Installation into /etc/inetd.conf is done with a line like this one:

```
3143  stream  tcp nowait  user /usr/sbin/tcpd
        /usr/local/sbin/in.acng /var/run/apt-cacher-ng/socket
```

Use "make in.acng" to compile the wrapper client.

Chapter 9: Contact

The planned features are listed in the file TODO. Don't hesitate to contact me if you really need something not found there and you can explain the severity of your request.

Please report any bugs found in apt-cacher (which are not obviously the complementary of missing features, see above ;-).