Alpine Linux package management

From Alpine Linux

Because Alpine Linux is designed to run from RAM, package management involves two phases:

- Installing / Upgrading / Deleting packages on a running system
- Restoring a system to a previously configured state (e.g. after reboot), including all previously installed packages and locally modified configuration files. **(RAM-Based Installs Only)**

apk is the tool used to install, upgrade, or delete software on a running sytem.
lbu is the tool used to capture the data necessary to restore a system to a previously configured state.

This page documents the apk tool (http://git.alpinelinux.org/cgit/apk-tools.git) - See the Alpine Local Backup page for the lbu tool.

Contents

Overview

The **apk** tool has the following applets:

add	Add new packages to the running system
del	Delete packages from the running system
fix	Attempt to repair or upgrade an installed package
update	Update the index of available packages
info	Prints information about installed or available packages
search	Search for packages or descriptions with wildcard patterns
upgrade	Upgrade the currently installed packages
cache	Maintenance operations for locally cached package repository
version	Compare version differences between installed and available packages
index	create a repository index from a list of packages
fetch	download (but not install) packages
audit	List changes to the file system from pristine package install state
verify	Verify a package signature

Packages and Repositories

Software packages for Alpine Linux are digitally signed tar.gz archives containing the programs, configuration files, and dependency metadata. They have the extension `.apk`, and are often called "a-packs"

The packages are stored in one or more *repositories*. A repository is simply a directory with a collection of *.apk files. The directory must include a special index file, named `APKINDEX.tar.gz` to be considered a repository.

The **apk** utility can install packages from multiple repositories. The list of repositories to check is stored in `/etc/apk/repositories`, one repository per line. If you booted from USB stick (`/media/sda1`) or CD-ROM (`/media/cdrom`), your repository file probably looks something like this:

Contents of `/etc/apk/repositories`

```
/media/sda1/apks/
```

In addition to local repositories, the **apk** utility uses **busybox wget** to fetch packages using *http:*, *https:* or *ftp:* protocols. The following is a valid repository file:

Contents of `/etc/apk/repositories`

```
/media/sda1/apks
http://dl-3.alpinelinux.org/alpine/v2.6/main
https://dl-3.alpinelinux.org/alpine/v2.6/main
ftp://dl-3.alpinelinux.org/alpine/v2.6/main
```

Note: Currently there are no public https or ftp repositories. The protocols are available for local repositories.

Repository pinning

You can specify additional "tagged" repositories in `/etc/apk/repositories`:

Contents of `/etc/apk/repositories`

```
http://nl.alpinelinux.org/alpine/v2.6/main
@edge http://nl.alpinelinux.org/alpine/edge/main
@testing http://nl.alpinelinux.org/alpine/edge/testing
```

After which you can "pin" dependencies to these tags using:

```
apk add stableapp newapp@edge bleedingapp@testing
```

Apk will now by default only use the untagged repositories, but adding a tag to specific package:

1. will prefer the repository with that tag for the named package, even if a later version of the package is available in another repository

2. *allows* pulling in dependencies for the tagged package from the tagged repository (though it *prefers* to use untagged repositories to satisfy dependencies if possible)

Update the Package list

Remote repositories change as packages are added and upgraded. To get the latest list of available packages, use the *update* command. The command downloads the `APKINDEX.tar.gz` from each repository and stores it in the local cache, typically `/var/cache/apk/`, `/var/lib/apk/` or `/etc/apk/cache/`.

```
apk update
```

Tip: If using remote repositories, it is a good idea to do an **update** just before doing an **add** or **upgrade** command. That way you know you are using the latest software available.

Add a Package

Use **add** to install packages from a repository. Any necessary dependencies are also installed. If you have multiple repositories, the **add** command installs the newest package.

```
apk add openssh
apk add openssh openntp vim
```

If you only have the main repository enabled in your configuration, apk will not include packages from the other repositories. To install a package from the edge/testing repository without changing your repository configuration file, use the command below. This will tell apk to use that particular repository.

```
apk add cherokee --update-cache --repository http://dl-3.alpinelinux.org/alpine/ed
```

Note: Be careful when using third-party or the testing repository. Your system can go down.

Remove a Package

Use **del** to remove a package (and dependencies that are no longer needed.)

```
apk del openssh
apk del openssh openntp vim
```

Upgrade a Running System

To upgrade *all* the packages of a running system, use **upgrade**:

```
apk update
apk upgrade
```

To upgrade *only a few* packages, use the **add** command with the *-u* or *--upgrade* option:

```
apk update
apk add --upgrade busybox
```

Note: Remember that when you reboot your machine, the remote repository will not be available until after networking is started. This means packages newer than your local boot media will likely not be installed after a reboot. To make an "upgrade" persist over a reboot, use a local cache.

Search for Packages

The **search** command searches the repository Index files for installable packages.

Examples:

- To list all packages available, along with their descriptions:

  ```
  apk search -v
  ```

- To list all packages are part of the ACF system:

  ```
  apk search -v 'acf*'
  ```

- To list all packages that list NTP as part of their description, use the *-d* or *--description* option:

  ```
  apk search -v --description 'NTP'
  ```

Info on Packages

The **info** command provides information on the contents of packages, their dependencies, and which files belong to a package.

For a given package, each element can be chosen (for example, *-w* to show just the webpage information); or all information is displayed with the *-a* command.

Example:

```
apk info -a zlib
```

```
zlib-1.2.5-r1 description:
A compression/decompression Library

zlib-1.2.5-r1 webpage:
http://zlib.net

zlib-1.2.5-r1 installed size:
94208

zlib-1.2.5-r1 depends on:
libc0.9.32

zlib-1.2.5-r1 is required by:
libcrypto1.0-1.0.0-r0
apk-tools-2.0.2-r4
openssh-client-5.4_p1-r2
openssh-5.4_p1-r2
libssl1.0-1.0.0-r0
freeswitch-1.0.6-r6
atop-1.25-r0

zlib-1.2.5-r1 contains:
lib/libz.so.1.2.5
lib/libz.so.1
lib/libz.so

zlib-1.2.5-r1 triggers:
```

As shown in the example you can determine

- The **description** of the package (*-d* or *--description*)
- The **webpage** where the application is hosted (*-w* or *--webpage*)
- The **size** the package will require once installed (in bytes) (*-s* or *--size*)
- What packages are required to use this one (**depends**) (*-R* or *--depends*)
- What packages require this one to be installed (**required by**) (*-r* or *--rdepends*)
- The **contents** of the package, that is, which files it installs (*-L* or *--contents*)
- Any **triggers** this package sets. (*-t* or *--triggers*) Listed here are directories that are watched; if a change happens to the directory, then the trigger script is run at the end of the apk add/delete. For example, doing a depmod once after installing all packages that add kernel modules.

Tip: The **info** command is also useful to determine which package a file belongs

to. For example:

```
apk info --who-owns /sbin/lbu
```

will display

```
/sbin/lbu is owned by alpine-conf-x.x-rx
```

Listing installed packages

To list all installed packages, use:

```
apk info
```

To list all installed packages in alphabetical order, with a description of each, do:

```
apk -vv info|sort
```

Additional apk Commands

In progress...

Local Cache

Alpine Linux needs to be able to pull packages from local media on boot. (You can't download packages from the net before you have a network connection.) Using remote repositories presents a problem. If the config files have been modified for a newer version of a package, and the older package is on local media, all sorts of fun can result.

The solution is a local cache of updated packages. This cache can be stored on any r/w media, typically the same location as the apkovl.

The cache is enabled by creating a symlink named */etc/apk/cache* that points to the cache directory.

To enable local cache run:

```
setup-apkcache
```

To enable Local Cache on releases prior v2.3

Alpine Linux version prior to v2.3 does not have the *setup-apkcache* tool so the symlink needs to be set up manually.

To manually enable Local Cache on HDD install

If you've installed Alpine to your hard drive (as 'sys'), then create a cache dir and then an */etc/apk/cache* symlink pointing to that dir:

```
mkdir -p /var/cache/apk
ln -s /var/cache/apk /etc/apk/cache
```

You normally don't need apk cache on HDD 'sys' installs but it might be handy if you re-install from net to have the packages cached.

To manually enable Local Cache on run-from-RAM installs

1. Create a **cache** directory on the device you store your lbu backups (typically, `/dev/sda1`.)

   ```
   mkdir /media/sda1/cache
   ```

Tip: If you get an error that says "mkdir: can't create directory '/media/usbdisk /cache': Read-only file system", then you probably need to remount your disk read-write temporarily. Try

```
mount -o remount,rw /media/sda1
```

and then don't forget to run

```
mount -o remount,ro /media/sda1
```

when you are done with the following commands

1. Create a symlink to this directory from `/etc/apk/cache`.

```
ln -s /media/sda1/cache /etc/apk/cache
```

2. Run an lbu commit to save the change (`/etc/apk/cache` is in `/etc` and is automatically backed up.)

```
lbu commit
```

3. Done. Now whenever you run an apk command that pulls a new package from a remote repository, the package is stored on your local media. On startup, Alpine Linux will check the local cache for new packages, and will install them if available.

Cache maintenance

Over time, newer packages will replace older ones; the cache directory will contain all older versions of packages.

Delete old packages

To clean out older versions of packages, run the **clean** command.

```
apk cache clean
```

or to see what is deleted

```
apk -v cache clean
```

Download missing packages

If you accidentally delete packages from the cache directory, you can make sure they are there with the **download** command,

```
apk cache download
```

Delete and download in one step

You can combine the two steps into one with the **sync** command - this cleans out old packages and downloads missing packages.

```
apk cache -v sync
```

Automatically Cleaning Cache on Reboot

To automatically attempt to validate your cache on reboot, you can add the above command to a `/etc/local.d/*.stop` file:

Contents of `/etc/local.d/cache.stop`

```
#!/bin/sh
# verify the local cache on shutdown
apk cache -v sync

# We should always return 0
return 0
```

Tip: Usually the only time you need to reboot is when things have gone horribly wrong; so this is a "best effort" to cover forgetting to sync the cache; It is much better to run **sync** immediately after adding or upgrading packages.

Note: Custom shutdown commands were formerly added to a `/etc/conf.d/local`; but that method is now deprecated.

Advanced APK Usage

Holding a specific package back

In certain cases, you may want to upgrade a system, but keep a specific package at a back level. It is possible to add "sticky" or versioned dependencies. For instance, to hold the *asterisk* package to the 1.6.2 level or lower:

```
apk add asterisk=1.6.0.21-r0
```

or

```
apk add 'asterisk<1.6.1'
```

after which a

```
apk upgrade
```

will upgrade the entire system, keeping the asterisk package at the 1.6.0 or lower level

To later upgrade to the current version,

```
apk add 'asterisk>1.6.1'
```

Retrieved from "http://wiki.alpinelinux.org/w/index.php?title=Alpine_Linux_package_management&oldid=11026"

Category: Package Manager

- This page was last modified on 1 July 2015, at 09:06.
-