Alpine local backup

From Alpine Linux

When you boot Alpine Linux in a run-from-RAM configuration, Alpine itself only loads a few required packages. But you probably want to do some personal adjustments (e.g., installing a package or doing some configuration). Since everything in RAM will get lost next time the box is rebooted or shut down, you will need to permanently save those modifications and adjustments. This is where `lbu` comes in handy!

Contents

Note: If you installed Alpine on HDD you don't need to use `lbu`. All your modifications have already been directly written to your HDD. However, `lbu` can still be useful for some maintenance tasks.

The first thing you need to know is this: **By default `lbu` only cares about modifications in `/etc` and its subfolders, with the exception of `/etc/init.d`!** Please have a look at lbu include to save files/folders located elsewhere than in `/etc`.

Alpine has the following tools for permanently storing your modifications:

- lbu
- lbu commit *(Same as 'lbu ci')*
- lbu package *(Same as 'lbu pkg')*
- lbu status *(Same as 'lbu st')*
- lbu list *(Same as 'lbu ls')*
- lbu diff

- lbu include *(Same as 'lbu inc' or 'lbu add')*
- lbu exclude *(Same as 'lbu ex' or 'lbu delete')*
- lbu list-backup *(Same as 'lbu lb')*
- lbu revert

In the below examples you will find some characters with special meaning:

- **| = or** *('lbu commit|ci' means that you can type ether 'lbu commit' or 'lbu ci')*
- **[ ] = optional** *(In 'lbu commit|ci [-nv]' you can just skip the '-n', '-v' or '-nv' part if you don't want it)*

Committing your changes

When you "commit" or save changes you've made to your system, `lbu` will generate a file named something like `myboxname.apkovl.tar.gz` *('myboxname' will be the same as the hostname)*. This file (that contains your modifications) is called your "apkovl". You will need to save your apkovl on some suitable media (floppy, usb, cf, other).

```
usage: lbu commit|ci [-nv] [<media>]

Options:
  -d    Remove old apk overlay files.
  -e    Protect configuration with a password.
  -n    Don't commit, just show what would have been committed.
  -p <password> Give encryption password on the command-line
  -v    Verbose mode.

The following values for <media> is supported: floppy usb
If <media> is not specified, the environment variable LBU_MEDIA will be used.

Password protection will use aes-256-cbc encryption. Other ciphers can be
used by setting the DEFAULT_CIPHER or ENCRYPTION environment variables.
For possible ciphers, try: openssl -v

The password used to encrypt the file, can either be specified with the -p
option or using the PASSWORD environment variable.

The environment variables can also be set in /etc/lbu/lbu.conf
```

Create a apkovl elsewhere than on your configured media

To "commit" changes, but overriding the destination of the generated apkovl file, use `lbu package` instead of `lbu commit`:

```
usage: lbu package|pkg -v [<dirname>|<filename>]

Options:
  -v   Verbose mode.

If <dirname> is a directory, a package named <hostname>.apkovl.tar.gz will
be created in the specified directory.

If <filename> is specified, and is not a directory, a package with the
specified name will be created.

If neither <dirname> nor <filename> is specified, a package named
<hostname>.apkovl.tar.gz will be created in current work directory.
```

Use SSH to create an apkovl on a different host

To create an apkovl of client on a centralized server, use `lbu package` with `-` as the apkovl name:

On the server:

```
ssh root@client "lbu package -" >client.apkovl.tar.gz
```

Check what will be added to your apkovl

`lbu status` lists what will be saved the next time you run `lbu commit`. Its default output is incremental, that is, to only show what files have changed since the last commit; but this can be overridden with the `-a` flag:

```
usage: lbu status|st [-av]

Options:
  -a    Compare all files, not just since last commit.
  -v    Also show include and exclude lists.
```

Another option is `lbu list`. This works like `lbu status -a` but the format of the output is a bit different. (It's strictly equivalent to `lbu package -v /dev/null`).

```
usage: lbu list|ls
```

A third option is `lbu diff`. This shows the same incremental changes that `lbu status` (without `-a`) does, but in a different format.

```
 usage: lbu diff
```

Include special files/folders to the apkovl

Assume that you have some files that you want to permanently save, but they are located somewhere else than in `/etc`.
It could be `/root/.ssh/authorized_keys` (used by `sshd` to authenticate ssh-users).
Such files/folders can be added to lbu's *include* list with the following command:

```
usage: lbu include|inc|add [-rv] <file> ...
       lbu include|inc|add [-v] -l

Options:
  -l    List contents of include list.
  -r    Remove specified file(s) from include list instead of adding.
  -v    Verbose mode.
```

Note: This information used to be maintained in `/etc/lbu/include`; now it's instead maintained together with the *exclude* list in `/etc/apk/protected_paths.d/lbu.list`. Either way, the command `lbu include` only modifies lbu's configuration; you will need to run `lbu commit` to actually create/modify your apkovl.

Exclude specific files/folders from the apkovl

Assume that you have some files located in `/etc` or one of its subfolders that you *do not* want to permanently save. It could be some log file or status file that for some reason isn't in `/var/log/` but in some location that would otherwise be tracked by `lbu`. Such files/folders can be added to lbu's *exclude* list by manually editing that file or using the following command:

```
usage: lbu exclude|ex|delete [-rv] <file> ...
       lbu exclude|ex|delete [-v] -l

Options:
  -l    List contents of exclude list.
  -r    Remove specified file(s) from exclude list instead of adding.
  -v    Verbose mode.
```

Note: As stated above, this information is now maintained in `/etc/apk/protected_paths.d/lbu.list`. The command `lbu exclude` also only modifies lbu's configuration; you will need to run `lbu commit` to actually create/modify your apkovl.

Execute a script as part of a backup

Sometimes it is necessary to run a script before or after a backup. Scripts in two optional directories allow for this:

```
/etc/lbu/pre-package.d
/etc/lbu/post-package.d
```

Files in those directories are run using run-script rules (meaning they must have the executable bit set, they are run in alphabetical order, and cannot contain an extension: `runme` works, but `runme.sh` does not.)

The scripts in `pre-package.d` are run before the apkovl is created; scripts in `post-package.d` are run after the apkovl is created.

Example

Rather than adding the raw database directories to `/etc/lbu/include`, you can do a "database dump". For purposes of example, we use `postgresql` `(https://pkgs.alpinelinux.org/package/main/x86_64/postgresql)`:

- Create `/etc/lbu/pre-package.d/sqldump` with the following contents:

```
pg_dumpall -U postgres | gzip -c >/root/pgdatabases.gz
```

- Mark the file executable: `chmod +x /etc/lbu/pre-package.d/sqldump`
- Create `/etc/lbu/post-package.d/sqldumpdelete` with the following contents:

```
rm -f /root/pgdatabases.gz
```

- Mark the file executable: `chmod +x /etc/lbu/post-package.d/sqldumpdelete`
- Finally, add the database dump file to the list of files to back up: `lbu include root/pgdatabases.gz`

Now whenever you do a `lbu commit`, the sql databases are dumped and gzipped to `/root/pgdatabases.gz`, and then the temporary file is deleted at the end of the lbu commit.

On a catastrophic restore, the databases are not automatically restored (that's not lbu's responsibility), but you will find a complete database dump in the `/root` directory, where it can be restored manually.

Multiple Backups

Lbu can now keep backups so you can revert to older, good known config. Set BACKUP_LIMIT in `/etc/lbu/lbu.conf` to the number of backups you want to keep.

You can list the current backups with:

```
lbu list-backup [<media>]
```

and you can revert to an older with:

```
lbu revert <filename> [<media>]
```

Nothing is written to your main system when "reverting"; this only affects which apkovl is considered active. If you've set BACKUP_LIMIT, then the previously active apkovl will be backed up before being overwritten.

See also

- Back Up a Flash Memory Installation
- Manually editing a existing apkovl

Retrieved from "http://wiki.alpinelinux.org/w/index.php?title=Alpine_local_backup&oldid=10328"

Categories: Installation | Storage | Booting | Package Manager

- This page was last modified on 22 January 2015, at 10:02.
- © Copyright 2008-2015 Alpine Linux Development Team all rights reserved