# Contributing

Termux is an open source application and it is built on users' contributions.

However, most of work is done by Termux maintainers on their spare time and

therefore only priority tasks are being completed.

Here are ways how you can help:

- [Fixing issues](#fixing-issues)

- [Hosting a mirror](#hosting-a-mirror)

- [Updating packages](#updating-packages)

Developer's wiki is available at https://github.com/termux/termux-packages/wiki.

## Fixing issues

Contribute to Termux by submitting new packages or fixing bugs. Pay attention to

[issues](https://github.com/termux/termux-packages/issues) labeled as

["bug report"](https://github.com/termux/termux-packages/issues?q=is%3Aopen+is%3Aissue+label%3A%22bug+report%22)

and ["help wanted"](https://github.com/termux/termux-packages/issues?q=is%3Aopen+is%3Aissue+label%3A%22help+wanted%22).

Note that issue solving may not be easy. If you decided to contribute to @termux,

ensure that you read the [developer's wiki](https://github.com/termux/termux-packages/wiki)

and you are ready to deal with errors on your own.

Pull requests containing incomplete work are not being merged.

### A note about package requests

We are rejecting package requests for single-file scripts and low-quality, infamous

and outdated, dead projects. We may also reject in packaging non-original, forked

projects.

Additionally we require that requested packages met these conditions:

- Package should have widely recognised OSS licenses like GNU GPL, MIT, Apache-2.0

and similar.

- Package should NOT be an outdated, dead project.

- Package should NOT be a part of language-specific ecosystem. Such packages are

installable through `pip`, `gem`, `cpan`, `npm`.

Please be ready that your package request will not be processed immediately.

## Hosting a mirror

Consider hosting own Termux repository mirror to help to distribute bandwidth usage

between different servers. That will reduce chance of hitting the traffic quota on

our [Bintray](https://bintray.com) account.

When your mirror is ready, open the issue so we can include it into our mirror

list.

### Server requirements

Since Bintray does not provide Rsync access to package repositories, you will need

to mirror it over HTTP. We recommend to use `apt-mirror` for that.

Here are the server requirements:

- 20 GB of disk free space.

- Stable network with 3+ TB of monthly bandwidth.

- Cron job for updating mirror at least once in 3 days.

It also preferrable that your server uses a caching CDN.

Here are the URLs for mirroring:

```

https://dl.bintray.com/termux/termux-packages-24

https://dl.bintray.com/grimler/game-packages-24

https://dl.bintray.com/grimler/science-packages-24

https://dl.bintray.com/grimler/termux-root-packages-24

https://dl.bintray.com/xeffyr/unstable-packages

https://dl.bintray.com/xeffyr/x11-packages

```

If you want to include your mirror into our mirror list, ensure that monthly server's

uptime is 80% or higher.

## Updating packages

Keeping packages up-to-date ensures that Termux users' will not experience the upstream

bugs or security issues and will be able to use the latest features.

Periodically check the [Repology](https://repology.org/projects/?inrepo=termux&outdated=1)

page to see what is outdated and submit a pull request with version update.

### How to update package

[![asciicast](https://asciinema.org/a/gVwMqf1bGbqrXmuILvxozy3IG.svg)](https://asciinema.org/a/gVwMqf1bGbqrXmuILvxozy3IG?autoplay=1&speed=2.0)

Most packages can be updated by just modifying variables `TERMUX\_PKG\_VERSION` and

`TERMUX\_PKG\_SHA256`.

- `TERMUX\_PKG\_VERSION`: a text field containing an original version of package.

- `TERMUX\_PKG\_SHA256`: a text field or an array of text fields containing SHA-256

checksum for each source code bundle defined by `TERMUX\_PKG\_SRCURL`.

More about `build.sh` variables you can read on [developer's wiki](https://github.com/termux/termux-packages/wiki/Creating-new-package#table-of-available-package-control-fields).

#### Rebuilding package with no version change

Changes to patch files and build configuration options require submission of a new

package release with a different version string. As we can't modify the original

package version, we append a number called \*revision\*. This number should be

incremented on each submitted build whenever project's version remains to be same.

Revision is specified through `TERMUX\_PKG\_REVISION` build.sh variable. To have

build.sh script easily readable, we require revision variable to be placed on

the next line after `TERMUX\_PKG\_VERSION`.

```

TERMUX\_PKG\_VERSION=1.0

TERMUX\_PKG\_REVISION=4

```

#### Downgrading a package or changing versioning scheme

Sometimes we need to downgrade a package or in any other way to change format of

version string but we also need to tell package manager that this is a new package

version which should be installed with `apt upgrade`. To force new build to be a

package update, we set a \*package epoch\*.

We don't have separate build.sh variable for specifying epoch, so we doing that

through `TERMUX\_PKG\_VERSION` variable. It takes following format:

```

${EPOCH}:${ORIG\_VERSION}

```

Epoch should be bumped on each change of versioning scheme or downgrade.

```

TERMUX\_PKG\_VERSION=1:0.5

TERMUX\_PKG\_REVISION=4

```

Note that if you are not @termux collaborator, pull request must contain a

\*description\* why you are submitting a package downgrade. All pull requests

which submit package downgrading without any serious reason will be denied.

#### Dealing with patch errors

Major changes introduced to packages often make current patches incompatible

with newer package version. Unfortunately, there no universal guide about

fixing patch issues as workaround is always based on changes introduced to

the new source code version.

Here are few things you may to try:

1. If patch fixing particular known upstream issue, check the project's VCS

for commits fixing the issue. There is a chance that patch is no longer

needed.

2. Inspecting the failed patch file and manually applying changes to source

code. Do so only if you understand the source code and changes introduced

by patch.

Regenerate patch file, e.g. with:

```

diff -uNr package-1.0 package-1.0.mod > previously-failed-patch-file.patch

```

Always check the CI (Github Actions) status for your pull request. If it fails,

then either fix or close it. Maintainers can fix it on their own, if issues are

minor. But they won't rewrite whole your submission.