Arch User Repository

The Arch User Repository (AUR) is a community-driven repository for Arch users. It contains package descriptions (**PKGBUILDs**) that allow you to compile a package from source with **makepkg** and then install it via **pacman**. The AUR was created to organize and share new packages from the community and to help expedite popular packages' inclusion into the **community** repository. This document explains how users can access and utilize the AUR.

A good number of new packages that enter the official repositories start in the AUR. In the AUR, users are able to contribute their own package builds (PKGBUILD and related files). The AUR community has the ability to vote for or against packages in the AUR. If a package becomes popular enough — provided it has a compatible license and good packaging technique — it may be entered into the *community* repository (directly accessible by **pacman** or **abs**).

Related articles

- **makepkg**
- **pacman**
- **PKGBUILD**
- **.SRCINFO**
- **AurJson**
- **AUR Trusted User Guidelines**
- **Official repositories**
- **Arch Build System**
- **Creating packages**
- **AUR helpers**

Contents

Getting started

Users can search and download PKGBUILDs from the **AUR Web Interface (https://aur.archlinux.org)**. These PKGBUILDs can be built into installable packages using **makepkg**, then installed using pacman.

- Ensure the `base-devel (https://www.archlinux.org/groups/x86_64/base-devel/)` package group is installed (`pacman -S --needed base-devel`).
- Glance over the **#FAQ** for answers to the most common questions.
- You may wish to adjust `/etc/makepkg.conf` to optimize for your processor prior to building packages from the AUR. A significant improvement in compile times can be realized on systems with multi-core processors by adjusting the MAKEFLAGS variable. Users can also enable hardware-specific optimizations in GCC via the CFLAGS variable. See **makepkg** for more information.

It is also possible to interact with the AUR through SSH: type `ssh aur@aur.archlinux.org help` for a list of available commands.

History

In the beginning, there was `ftp://ftp.archlinux.org/incoming` , and people contributed by simply uploading the PKGBUILD, the needed supplementary files, and the built package itself to the server. The package and associated files remained there until a **Package Maintainer** saw the program and adopted it.

Then the Trusted User Repositories were born. Certain individuals in the community were allowed to host their own repositories for anyone to use. The AUR expanded on this basis, with the aim of making it both more flexible and more usable. In fact, the AUR maintainers are still referred to as TUs (Trusted Users).

Between 2015-06-08 and 2015-08-08 the AUR transitioned from version 3.5.1 to 4.0.0, introducing the use of Git repositories for publishing the PKGBUILDs. Existing packages were dropped unless manually migrated to the new infrastructure by their maintainers.

Git repositories for AUR3 packages

The **AUR Archive (https://github.com/aur-archive)** on GitHub has a repository for every package that was in AUR 3 at the time of the migration. Alternatively, there is the **aur3-mirror (https://github.com/felixonmars/aur3-mirror/)** repository which provides the same.

Searching

The AUR web interface can be found at **https://aur.archlinux.org/**, and an interface suitable for accessing the AUR from a script can be found at **https://aur.archlinux.org/rpc.php**.

Queries search package names and descriptions via a MySQL LIKE comparison. This allows for more flexible search criteria (e.g. try searching for `tool%like%grep` instead of `tool like grep`). If you need to search for a description that contains `%`, escape it with `\%`.

Installing packages

Installing packages from the AUR is a relatively simple process. Essentially:

1. Acquire the build files, including the **PKGBUILD** and possibly other required files, like **systemd** units and patches (often not the actual code).
2. Verify that the **PKGBUILD** and accompanying files are not malicious or untrustworthy.
3. Run `makepkg -si` in the directory where the files are saved. This will download the code, resolve the dependencies with **pacman**, compile it, package it, and install the package.

Note: The AUR is unsupported, so any packages you install are *your responsibility* to update, not pacman's. If packages in the official repositories are updated, you will need to rebuild any AUR packages that depend on those libraries.

Prerequisites

First ensure that the necessary tools are installed by **installing** the **`base-devel (https://www.archlinux.org/groups/x86_64/base-devel/)`** group which includes **`make (https://www.archlinux.org/packages/?name=make)`** and other tools needed for compiling from source.

Note: Packages in the AUR assume that the **`base-devel (https://www.archlinux.org/groups/x86_64/base-devel/)`** group is installed, i.e. they do not list the group's members as dependencies explicitly.

Next choose an appropriate build directory. A build directory is simply a directory where the package will be made or "built" and can be any directory. The examples in the following sections will use `~/builds` as the build directory.

Acquire build files

Locate the package in the AUR. This is done using the search feature (text field at the top of the **AUR home page (https://aur.archlinux.org/)**). Clicking the application's name in the search list brings up an information page on the package. Read through the description to confirm that this is the desired package, note when the package was last updated, and read any comments.

There are several methods for acquiring the build files:

- Clone the **git** repository that is labeled as the "Git Clone URL" in the "Package Details":

```
$ git clone https://aur.archlinux.org/package_name.git
```

  An advantage of this method is that you can easily get updates to the package via `git pull`.

- Download the build files with your web browser by clicking the "Download snapshot" link under "Package Actions" on the right hand side. This will download a compressed file, which must be extracted (preferably in a directory set aside for AUR builds)

```
$ tar -xvf package_name.tar.gz
```

- Similarly, you can download a tarball from the terminal (and extract it):

```
$ curl -L -O https://aur.archlinux.org/cgit/aur.git/snapshot/package_name.tar.gz
```

Build and install the package

Change directories to the directory containing the package's **PKGBUILD**.

Warning: Carefully check all files. Carefully check the `PKGBUILD` and any *.install* file for malicious commands. `PKGBUILD`s are **bash** scripts containing functions to be executed by *makepkg*: these functions can contain *any* valid commands or Bash syntax, so it is totally possible for a `PKGBUILD` to contain dangerous commands through malice or ignorance on the part of the author. Since *makepkg* uses *fakeroot* (and should never be run as root), there is some level of protection but you should never count on it. If in doubt, do not build the package and seek advice on the forums or mailing list.

```
$ cd package_name
$ less PKGBUILD
$ less package_name.install
```

Make the package. After manually confirming the integrity of the files, run **makepkg** as a normal user:

```
$ makepkg -si
```

- `-s` / `--syncdeps` automatically resolves and installs any dependencies with **pacman** before building. If the package depends on other AUR packages, you will need to manually install them first.
- `-i` / `--install` installs the package if it is built successfully. Alternatively the built package can be installed with `pacman -U *package*.pkg.tar.xz`.

Other useful flags are

- `-r` / `--rmdeps` removes build-time dependencies after the build, as they are no longer needed. However these dependencies may need to be reinstalled the next time the package is updated.
- `-c` / `--clean` cleans up temporary build files after the build, as they are no longer needed. These files are usually needed only when debugging the build process.

Note: The above example is only a brief summary of the build process. It is **highly** recommended to read the **makepkg** and **ABS** articles for more details.

Feedback

The **AUR Web Interface (https://aur.archlinux.org)** has a comments facility that allows users to provide suggestions and feedback on improvements to the PKGBUILD contributor. Avoid pasting patches or PKGBUILDs into the comments section: they quickly become obsolete and just end up needlessly taking up lots of space. Instead email those files to the maintainer, or even use a **pastebin**.

One of the easiest activities for **all** Arch users is to browse the AUR and **vote** for their favourite packages using the online interface. All packages are eligible for adoption by a TU for inclusion in the **community** repository, and the vote count is one of the considerations in that process; it is in everyone's interest to vote!

Sharing and maintaining packages

Note: Please see **Talk:Arch User Repository#Scope of the AUR4 section (https://wiki.archlinux.org/index.php?title=Talk:Arch_User_Repository&oldid=484964#Scope_of_the_AUR4_section)** before making changes to this section.

Users can **share** PKGBUILDs using the Arch User Repository. It does not contain any binary packages but allows users to upload PKGBUILDs that can be downloaded by others. These PKGBUILDs are completely unofficial and have not been thoroughly vetted, so they should be used at your own risk.

Submitting packages

Warning: Before attempting to submit a package you are expected to familiarize yourself with **Arch packaging standards** and all the articles under "Related articles". **Verify carefully** that what you are uploading is correct. Packages that violate the rules may be **deleted** without warning.

If you are unsure in any way about the package or the build/submission process even after reading this section twice, submit the PKGBUILD to the **AUR mailing list (https://mailman.archlinux.org/mailman/listinfo/aur-general)**, the **AUR forum (https://bbs.archlinux.org/viewforum.php?id=4)** on the Arch forums, or ask on our **IRC channel** for public review before adding it to the AUR.

Rules of submission

When submitting a package to the AUR, observe the following rules:

- The submitted PKGBUILDs must not build applications **already in any** of the **official** binary **repositories** under any circumstances. Check the **official package database (https://www.archlinux.org/packages/)** for the package. If any version of it exists, **do not** submit the package. If the official package is out-of-date, flag it as such. If the official package is broken or is lacking a feature, then please file a **bug report (https://bugs.archlinux.org/)**.

  Exception to this strict rule may only be packages having **extra features** enabled and/or **patches** in comparison to the official ones. In such an occasion the `pkgname` should be different to express that difference. For example, a package for GNU screen containing the sidebar patch could be named `screen-sidebar`. Additionally the `provides=('screen')` array should be used in order to avoid conflicts with the official package.

- **Check the AUR** if the package **already exists**. If it is currently maintained, changes can be submitted in a comment for the maintainer's attention. If it is unmaintained or the maintainer is unresponsive, the package can be adopted and updated as required. Do not create duplicate packages.

- Make sure the package you want to upload is **useful**. Will anyone else want to use this package? Is it extremely specialized? If more than a few people would find this package useful, it is appropriate for submission.

  The AUR and official repositories are intended for packages which install generally software and software-related content, including one or more of the following: executable(s); config file(s); online or offline documentation for specific software or the Arch Linux distribution as a whole; media intended to be used directly by software.

- Do not use `replaces` in an AUR PKGBUILD unless the package is to be renamed, for example when *Ethereal* became *Wireshark*. If the package is an **alternate version of an already existing package**, use `conflicts` (and `provides` if that package is required by others). The main difference is: after syncing (-Sy) pacman immediately wants to replace an installed, 'offending' package upon encountering a package with the matching `replaces` anywhere in its repositories; `conflicts`, on the other hand, is only evaluated when actually installing the package, which is usually the desired behavior because it is less invasive.

- Submitting **binaries** should be **avoided** if the sources are available. The AUR should not contain the binary tarball created by makepkg, nor should it contain the filelist.

- Please add a **comment line** to the top of the `PKGBUILD` file which contains information about the current **maintainers** and previous **contributors**, respecting the following format. Remember to disguise your email to protect against spam. Additional or unneeded lines are facultative.

If you are assuming the role of maintainer for an existing PKGBUILD, add your name to the top like this

```
# Maintainer: Your Name <address at domain dot tld>
```

If there were previous maintainers, put them as contributors. The same applies for the original submitter if this is not you. If you are a co-maintainer, add the names of the other current maintainers as well.

```
# Maintainer: Your name <address at domain dot tld>
# Maintainer: Other maintainer's name <address at domain dot tld>
# Contributor: Previous maintainer's name <address at domain dot tld>
# Contributor: Original submitter's name <address at domain dot tld>
```

Authentication

For write access to the AUR, you need to have an **SSH key pair**. The content of the public key needs to be copied to your profile in *My Account*, and the corresponding private key configured for the `aur.archlinux.org` host. For example:

```
~/.ssh/config
```

```
Host aur.archlinux.org
  IdentityFile ~/.ssh/aur
  User aur
```

You should **create a new key pair** rather than use an existing one, so that you can selectively revoke the keys should something happen:

```
$ ssh-keygen -f ~/.ssh/aur
```

Tip: You can add multiple public keys to your profile by separating them with a newline in the input field.

Creating a new package

In order to create a new, empty, local Git repository for a package, simply `git clone` the remote repository with the corresponding name. If the package does not exist on AUR yet, you will see the following warning:

```
$ git clone ssh://aur@aur.archlinux.org/package_name.git
```

```
Cloning into 'package_name'...
warning: You appear to have cloned an empty repository.
Checking connectivity... done.
```

Note: When a package on the AUR is deleted, the git repository is not deleted so it is possible for a deleted package's repository to not be empty when cloned if someone is creating a package with the same name.

If you have already created a git repository, you can simply create a remote for the AUR git repository and then fetch it:

```
$ git remote add remote_name ssh://aur@aur.archlinux.org/package_name.git
$ git fetch remote_name
```

where `remote_name` is the name of the remote to create (*e.g.,* "origin"). See **Git#Using remotes** for more information.

The new package will appear on AUR after you *push* the first commit. You can now add the source files to the local copy of the Git repository. See **#Uploading packages**.

Warning: Your AUR commits will be authored according to your git user name and email address and it is very difficult to change commits after you push them (see **FS#45425 (https://bugs.archlinux.org/task/45425)**). If you want to push to AUR under a different name/email, you can change them for this package via `git config user.name [...]` and `git config user.email [...]`. Review your commits before pushing them!

Uploading packages

The procedure for uploading packages to the AUR is the same for new packages and package updates. You need at least **PKGBUILD** and **.SRCINFO** in the top-level directory to *push* your package to AUR.

Note: You need to regenerate the `.SRCINFO` every time you change `PKGBUILD` metadata, such as **pkgver()** updates. Otherwise the AUR will not show the updated version numbers.

To upload, add the `PKGBUILD`, `.SRCINFO`, and any helper files (like *.install* files or local source files like *.patch*) to the *staging area* with `git add`, commit them to your local tree with a commit message with `git commit`, and finally publish the changes to the AUR with `git push`.

For example:

```
$ makepkg --printsrcinfo > .SRCINFO
$ git add PKGBUILD .SRCINFO
$ git commit -m "useful commit message"
$ git push
```

Tip:

- If you initially forgot to commit the `.SRCINFO` and added it in a later commit, the AUR will still reject your pushes because the `.SRCINFO` must exist for *every* commit. To solve this problem you can use **git rebase (https://git-scm.com/docs/git-rebase)**

with the `--root` option or **git filter-branch (https://git-scm.com/docs/git-filter-branch)** with the `--tree-filter` option.
- To prevent untracked files from commits and to keep the working directory as clean as possible, exclude all files with `.gitignore` and force-add files instead. See **dotfiles#Using gitignore**.

Maintaining packages

- Check for feedback and comments from other users and try to incorporate any improvements they suggest; consider it a learning process!
- Please do not leave a comment containing the version number every time you update the package. This keeps the comment section usable for valuable content mentioned above. **AUR helpers** are suited better to check for updates.
- Please do not just submit and forget about packages! It is the maintainer's job to maintain the package by checking for updates and improving the PKGBUILD.
- If you do not want to continue to maintain the package for some reason, `disown` the package using the AUR web interface and/or post a message to the AUR Mailing List. If all maintainers of an AUR package disown it, it will become an **"orphaned" (https://aur.archlinux.org/packages/?O=0&SeB=nd&K=&outdated=&SB=n&SO=a&PP=50&do_Orphans=Orphans)** package.

Other requests

Orphan, deletion and merge requests can be created by clicking on the "Submit Request" link under "Package Actions" on the right hand side. This automatically sends a notification email to the current package maintainer and to the **aur-requests mailing list (https://mailman.archlinux.org/mailman/listinfo/aur-requests)** for discussion. **Trusted Users** will then either accept or reject the request.

- Orphan requests will be granted after two weeks if the current maintainer did not react.
- Merge requests are to delete the package base and transfer its votes and comments to another package base. The name of the package base to merge into is required. Note this has nothing to do with 'git merge' or GitHub's merge requests.
- Deletion requests require the following information:
  - A short note explaining the reason for deletion. Note that a package's comments does not sufficiently point out the reasons why a package is up for deletion. Because as soon as a TU takes action, the only place where such information can be obtained is the aur-requests mailing list.
  - Supporting details, like when a package is provided by another package, if you are the maintainer yourself, it is renamed and the original owner agreed, etc.
  - Deletion requests can be rejected, in which case if you are the maintainer or the package you will likely be advised to disown the package to allow adoption by another packager.

Web interface translation

See **i18n.txt (https://projects.archlinux.org/aurweb.git/tree/doc/i18n.txt)** in the AUR source tree for information about creating and maintaining translation of the AUR web interface.

Comment syntax

The **Python-Markdown (https://python-markdown.github.io/)** syntax is supported in comments. It provides basic **Markdown** syntax to format comments. Note this implementation has some occasional **differences (https://python-markdown.github.io/#differences)** with the official **syntax rules (https://daringfireball.net/projects/markdown/syntax)**. Commit hashes to the Git repository of the package and references to Flyspray tickets are converted to links automatically. Long comments are collapsed and can be expanded on demand.

FAQ

What is the AUR?

The AUR (Arch User Repository) is a place where the Arch Linux community can upload **PKGBUILDs** of applications, libraries, etc., and share them with the entire community. Fellow users can then vote for their favorites to be moved into the **community** repository to be shared with Arch Linux users in binary form.

What kind of packages are permitted on the AUR?

The packages on the AUR are merely "build scripts", i.e. recipes to build binaries for pacman. For most cases, everything is permitted, subject to the abovementioned **usefulness and scope guidelines**, as long as you are in compliance with the licensing terms of the content. For other cases, where it is mentioned that "you may not link" to downloads, i.e. contents that are not redistributable, you may only use the file name itself as the source. This means and requires that users already have the restricted source in the build directory prior to building the package. When in doubt, ask.

How can I vote for packages in the AUR?

Sign up on the **AUR website (https://aur.archlinux.org/)** to get a "Vote for this package" option while browsing packages. After signing up it is also possible to vote from the commandline with `aurvote (https://aur.archlinux.org/packages/aurvote/)`AUR, `aurvote-git (https://aur.archlinux.org/packages/aurvote-git/)`AUR or `aur-auto-vote-git (https://aur.archlinux.org/packages/aur-auto-vote-git/)`AUR.

Alternatively, if you have set up **ssh authentication** as above, you can directly vote from the command line using your ssh key. This means that you won't need to save or type in your AUR password.

```
ssh aur@aur.archlinux.org vote <PACKAGE_NAME>
```

What is a Trusted User / TU?

A **Trusted User**, in short TU, is a person who is chosen to oversee AUR and the **community** repository. They are the ones who maintain popular PKGBUILDs in *community*, and overall keep the AUR running.

What is the difference between the Arch User Repository and the community repository?

The Arch User Repository is where all PKGBUILDs that users submit are stored, and must be built manually with **makepkg**. When PKGBUILDs receive enough community interest and the support of a TU, they are moved into the **community** repository (maintained by the TUs), where the binary packages can be installed with **pacman**.

Foo in the AUR is outdated; what should I do?

First, you should flag the package *out-of-date* indicating details on why the package is outdated, preferably including links to the release announcement or the new release tarball. You should also try to reach out to the maintainer directly by email. If there is no response from the maintainer after *two weeks*, you can file an *orphan* request. This means you ask a **Trusted User** to disown the package base. This is to be done only if the package requires maintainer action, that he/she is not responding and you already tried to contact him/her previously.

In the meantime, you can try updating the package yourself by editing the PKGBUILD locally. Sometimes, updates do not require changes to the build or package process, in which case simply updating the `pkgver` or `source` array is sufficient.

Note: **VCS packages** are not considered out of date when the pkgver changes, do not flag them as the maintainer will merely unflag the package and ignore you. AUR maintainers should not commit mere pkgver bumps.

Foo in the AUR does not compile when I run makepkg; what should I do?

You are probably missing something trivial.

1. **Upgrade the system** before compiling anything with `makepkg` as the problem may be that your system is not up-to-date.
2. Ensure you have both **base (https://www.archlinux.org/groups/x86_64/base/)** and **base-devel (https://www.archlinux.org/groups/x86_64/base-devel/)** groups installed.
3. Try using the `-s` option with `makepkg` to check and install all the dependencies needed before starting the build process.

Be sure to first read the PKGBUILD and the comments on the AUR page of the package in question. The reason might not be trivial after all. Custom CFLAGS, LDFLAGS and MAKEFLAGS can cause failures. It is also possible that the PKGBUILD is broken for everyone. If you cannot figure it out on your own, just report it to the maintainer e.g. by posting the errors you are getting in the comments on the AUR page.

How do I create a PKGBUILD?

The best resource is the wiki page about **creating packages**. Remember to look in AUR before creating the PKGBUILD as to not duplicate efforts.

I have a PKGBUILD I would like to submit; can someone check it to see if there are any errors?

If you would like to have your PKGBUILD reviewed, post it on the **aur-general mailing list (https://mailman.archlinux.org/mailman/listinfo/aur-general)** to get feedback from the TUs and fellow AUR members. You could also get help from the **IRC channel**, #archlinux-aur on irc.freenode.net. You can also use **namcap** to check your PKGBUILD and the resulting package for errors.

How to get a PKGBUILD into the community repository?

Usually, at least 10 votes are required for something to move into **community**. However, if a TU wants to support a package, it will often be found in the repository.

Reaching the required minimum of votes is not the only requirement, there has to be a TU willing to maintain the package. TUs are not required to move a package into the *community* repository even if it has thousands of votes.

Usually when a very popular package stays in the AUR it is because:

- Arch Linux already has another version of a package in the repositories
- The package is AUR-centric (e.g. an **AUR helper**)
- Its license prohibits redistribution

See also **DeveloperWiki:Community repo candidates** and **Rules for Packages Entering the community Repo**.

How can I speed up repeated build processes?

If you frequently compile code that uses GCC - say, a Git or SVN package - you may find **ccache**, short for "compiler cache", useful.

What is the difference between foo and foo-git packages?

Many AUR packages are presented in regular ("stable") and development versions ("unstable"). A development package usually has a suffix such as `-cvs`, `-svn`, `-git`, `-hg`, `-bzr` or `-darcs`. While development packages are not intended for regular use, they may offer new features or bugfixes. Because these packages download the latest available source when you execute `makepkg`, a package version to track possible updates is not directly available for these. Likewise, these packages cannot perform an authenticity checksum, instead it is relied on the maintainer(s) of the Git repository.

See also **System maintenance#Use proven software packages**.

Why has foo disappeared from the AUR?

It is possible the package has been adopted by a TU and is now in the **community** repository.

Packages may be deleted if they did not fulfill the **#Rules of submission**. See the **aur-requests archives (https://lists.archlinux.org/pipermail/aur-requests/)** for the reason for deletion.

If the package used to exist in AUR3, it might not have been **migrated to AUR4 (https://lists.archlinux.org/pipermail/aur-general/2015-August/031322.html)**. See the **#Git repositories for AUR3 packages** where these are preserved.

How do I find out if any of my installed packages disappeared from AUR?

The simplest way is to check the HTTP status of the package's AUR page:

```
$ comm -23 <(pacman -Qqm | sort) <(curl https://aur.archlinux.org/packages.gz | gzip -cd | sor
t)
```

If you use an **AUR helper**, you can shorten this script by replacing the curl command with whatever command queries the AUR for a package.

How can I obtain a list of all AUR packages?

- **https://aur.archlinux.org/packages.gz**
- Use `aurpkglist` from **`python3-aur (https://aur.archlinux.org/packages/python3-aur/)`**AUR

See also

- **AUR Web Interface (https://aur.archlinux.org)**
- **AUR Mailing List (https://lists.archlinux.org/listinfo/aur-general)**
- **DeveloperWiki:AUR Cleanup Day**

Retrieved from "https://wiki.archlinux.org/index.php?title=Arch_User_Repository&oldid=514483"

- This page was last edited on 23 March 2018, at 04:30.
-