# Using Bazaar With Launchpad

***Release 2.8.0dev1***

**Bazaar Developers**

August 11, 2021

## Contents

# 1 Motivation

## 1.1 Communities are different to teams

The team of people required to create the initial release of a piece of software may vary in size from one person to several thousand people. Depending on the requirements, the challenges involved, both technical and managerial, can be immense. As explained in the Bazaar User Guide, selecting "just right" processes and using tools like Bazaar to support matching workflows can greatly help.

Success with software though requires more than a great team - it requires a healthy, active *community*. This group is typically far larger than the team as it includes everyone interested in the software: the team, users, training partners, support partners, third-party developers and so on.

Great communities are well understood in the free software community. Their applicability extends well beyond that though: most successful commercial software vendors are well skilled at building and managing the communities that grow up around their flagship products.

Like great teams, great communities don't just happen. Good policies and guidelines are essential for fostering the right sort of behaviour and healthy relationships between participants. For a deeper look at this topic, see Karl Fogel's landmark book: Producing Open Source Software.

## 1.2 The need for Collaborative Development Environments

An intelligent toolset is also important for tracking and managing community information and workflows. These tools are called Collaborative Development Environments (CDEs). These toolsets are typically web-based and manage things such as announcements, issues/bugs, questions and answers, downloads, documents and source code. Some examples of CDEs include Launchpad, SourceForge, java.net and SAP Community Network.

## 1.3 Helping communities work with related communities

Many successful products have a huge number of downstream dependencies. In other words, a new challenge arises with success: dealing with other communities and understanding how your changes will impact them. This is most obvious for projects like:

- software languages, e.g. Python, PHP, Ruby, Java, Perl, etc.
- compilers, e.g. gcc, JDK, etc.
- libraries, e.g. zlib, openssl, etc.
- frameworks, e.g. Zope, Ruby on Rails, Spring, etc.

However it applies equally for popular applications on which add-ons are built, e.g. Firefox, Thunderbird, OpenOffice.org, Drupal, Wordpress, Joomla, etc.

Tools that assist communities work together to track and manage issues and fixes across community boundaries are required. These tools help people at both ends of the spectrum:

- users can report problems in their terms, e.g. rendering of image type X is broken in application Y on operating system Z

- developers can better appreciate the downstream impact of making a change or fix, e.g. fixing this bug in a graphics library will make the users of these 5 applications on these 10 operating systems happy.

People in the middle play the essential role of *joining the dots* and communicating up and down the line. In many cases, they may also fix the problem for end users, releasing a patch and pushing a suggested fix to the upstream development team. Keeping track of all that over time in a sustainable way is no easy task.

### 1.4 Launchpad: More development, less friction

As well as sponsoring Ubuntu and Bazaar development, Canonical provides Launchpad, <https://launchpad.net/>, as a free service for the community. Launchpad is one of the most exciting CDEs around for several notable reasons:

- it models relationships between many of things tracked, e.g. source code branches can be associated with bug fixes
- as well as managing historical knowledge, it supports future development planning and tracking by providing features such as roadmaps, milestones and blueprints
- it provides translation tools and packaging services so that barriers are reduced for translators and testers wishing to join your community and help out
- it provides a nexus for different communities to work together on related issues and roadmaps.

In other words, Launchpad has been designed to help your community grow and to reduce the workflow friction both *within* your community and *between* communities. Ultimately, that means less time on mechanical tasks and more time for interesting development.

### 1.5 Bazaar: Launchpad's VCS client

This tutorial looks at how Bazaar and Launchpad can be used together and how they complement each other. It is important to remember that:

1. Bazaar can be used without Launchpad
2. Launchpad can be used without Bazaar.

By design though, their sum is greater than the individual parts.

## 2 Finding and browsing branches using Launchpad

### 2.1 Finding available branches

While there are many advantages in adopting distributed version control, one of the things that disappears is the all-knowing central server with knowledge about all available branches. Indeed in a distributed environment, interesting branches can literally exist in 100s of locations across the Internet (or within an Intranet for that matter).

Launchpad fills this gap by providing a registry of branches.

### 2.2 Registering branches

Branches can be uploaded to Launchpad or simply registered as being available in an external location. Branches can also be given a Status such as *New*, *Development*, *Mature* or *Abandoned*.

Note: External branches can even be hosted in legacy version control tools, i.e. CVS and Subversion. Code in these systems will be scanned and converted to Bazaar branches on a periodic basis. For maximum fidelity of course, it is preferable for external branches to be hosted in Bazaar.

### 2.3 Browsing branches

Branches can be listed, filtered and sorted by numerous attributes including Name, Registrant, Author, Status, Age and time of last commit. Browsing of branches is also provided making it easy to see things such as:

- where the branch can be downloaded from
- how to upload changes
- recent commits and the changes made by each
- the source code of individual files for a given version.

## 3 Accessing code in Launchpad using Bazaar

### 3.1 Getting the code for a project

As Launchpad keeps track of thousands of projects and their latest code whether it be managed by Bazaar, CVS or Subversion, Bazaar users can grab that code as easily as this:

```
bzr branch lp:project-name
```

where *project-name* is the Launchpad project ID. Here are some examples:

```
bzr branch lp:inkscape
bzr branch lp:amarok
bzr branch lp:python
bzr branch lp:rails
bzr branch lp:java-gnome
```

You can then browse the code locally using your favorite editor or IDE and change the code if you wish.

If a project has multiple series registered (e.g. a development series and a maintenance series), the latest code for a given series can be fetched using:

```
bzr branch lp:project-name/series
```

### 3.2 Publishing your changes

Having fixed that annoying bug or added that cool feature you've always wanted, it's time to impress your friends and make the world a better place by making your code available to others. As explained earlier, Launchpad is a free Bazaar code hosting service so you can push your branch to it and others can access your code from there. For example, assuming you are a member of the relevant team, login to launchpad like this:

```
bzr launchpad-login userid
```

where *userid* is your Launchpad user ID. You can then push your changes to a team branch like this:

```
bzr push lp:~team-name/project-name/branch-name
```

Others can then download your code like this:

```
bzr branch lp:~team-name/project-name/branch-name
```

## 3.3 Personal branches

Even if you are not a member of a team, Launchpad can be used to publish your changes. In this case, simply create a personal branch like this:

```
bzr push lp:~userid/project-name/branch-name
```

Others can then download your code like this:

```
bzr branch lp:~userid/project-name/branch-name
```

Note: Even when publishing to a personal branch, it is polite to notify the upstream developers about your branch so they can pull your changes from it if they are generally applicable to all users and meet the project's quality standards.

## 3.4 Package source branches

When maintaining packages for Ubuntu using Bazaar you can easily access the package's source branch on Launchpad. The package's source branch in the current (default) series can be downloaded like this:

```
bzr branch ubuntu:package
```

where *package* is the name of the Ubuntu package you want to access. To download the package branch for a specific series in Ubuntu (e.g. Maverick or Lucid), use this:

```
bzr branch ubuntu:maverick/package
```

Ubuntu distroseries can also be abbreviated to just their first letter. For example, the above could also be written:

```
bzr branch ubuntu:m/package
```

You can also download the package source branch from Launchpad for several Debian series. The default series can be downloaded like this:

```
bzr branch debianlp:package
```

and a specific series can be downloaded like this:

```
bzr branch debianlp:lenny/package
```

Note that the `debianlp:` scheme access the Debian source branch for a package from Launchpad only.

# 4 Linking branches using Launchpad

## 4.1 Associating a branch with a bug

After registering a branch, you can associate it to a bug so that people interested in that bug can track and download the fix as it becomes available.

To do this, the steps are:

1. Navigate to the bug in question.
2. Select *Add branch* under *Actions*.

3. Select the branch.
4. Optionally set the State of the relationship. This is *Fix In Progress* by default but you may wish to set it to another state such as *Fix Available* if the branch already addresses the issue.

If you wish, you can also provide some arbitrary comments about the relationship between the bug and the branch.

## 4.2 Changing the state in Launchpad while committing in Bazaar

Bazaar and Launchpad can work together to reduce some of the status housekeeping for you. When you commit using Bazaar, use the –fixes option like this:

```
bzr commit --fixes lp:1234 -m "..."
```

where 1234 is the bug ID. This will changes the State of the bug-branch relationship to *Fix Available*. If the one commit fixes multiple issues, the –fixes option can be specified multiple times.

One of the cool things about this feature is that Launchpad does not need to be accessible when making the commit. The `--fixes` option works by storing metadata which Launchpad will detect next time the branch is pushed to it or scanned once online again.

Note: Launchpad will not implicitly close a bug just because a branch is available that fixes it. There are several reasons for this. Firstly, the branch usually needs to be merged into the trunk (main development branch) before most teams consider it fixed. Secondly, many teams have a separate process for confirming bugs are fixed over and above a developer saying so.

As explained later, merge control features are currently under development in Launchpad and automatically changing the status of bugs to *Fix Committed* will be more appropriate once those features are in place.

## 4.3 Associating a branch with a blueprint

After registering a branch, you can associate it to a blueprint so that people interested in that blueprint can track and test the feature as it develops.

To do this, the steps are:

1. Navigate to the blueprint in question.
2. Select *Link branch* under *Actions*.
3. Select the branch.

If you wish, you can also provide some arbitrary comments about the relationship between the blueprint and the branch.

# 5 Managing releases using Launchpad

## 5.1 Integrating changes

Once a branch has been developed and published, communities typically go through a rigorous process before those changes are integrated into the core product and rolled out to end users. Some of the steps involved may include:

- peer review of the changes
- deciding which releases to include the changes in, e.g. the next maintenance release, the next major release, or both

- running functional regression tests
- benchmarking to ensure performance remains acceptable
- packaging into early access releases for end user testing
- documentation updates, e.g. Release Notes for the targeted releases
- translation of the user interface and documentation into multiple languages.

This section briefly looks at some of the features in Launchpad that help get good quality code into production. Strong integration with Bazaar is core to making this happen smoothly.

Note: Where indicated, some of the features below are still under development. If one or more of these features interest you, please consider joining the Launchpad beta test team at this link: https://help.launchpad.net/JoiningLaunchpadBetaTesters. You can then get early access to features and provide feedback to the developers before wider roll-out.

## 5.2 Branch merge proposals

After navigating to a branch in Launchpad, one of the available actions is *Propose for merging*. This lets you nominate which branch this code ought to be merged into.

Tracking the knowledge about which branches are proposed to be merged into a codeline helps Release Managers keep on top of what still needs to be completed, or can be completed, before a ship date. Using this information, they can ensure branches are merged after completing any necessary reviews. In the simple case, the Release Manager may manually merge branches. In more advanced cases, the merging could be automatically done by a robot (like PQM) when the branch reaches the right state (e.g. *Review completed*).

## 5.3 Code review tracking

A number of features are under development in Launchpad to track the states, conversations and outcomes of code reviews. These features are expected to be integrated with branch merge proposals and branch browsing features.

## 5.4 Personal Package Archives (PPAs)

PPAs help developers and development teams get custom builds into the hands of users for early testing and feedback. In other words, a PPA allows a developer to form a community of testers who are interested in their changes. The testing community can install the packages, run them for the test period and then remove them cleanly from their system.

See https://help.launchpad.net/PPAQuickStart for further details.

## 5.5 Translations

The Translations module in Launchpad is designed to make it easy for anyone to get involved translating applications to languages they know. Translators are shielded from the low level details.

Launchpad keeps track of the translations for each major version of a project separately, allowing translators to continue to improve the translations of your stable releases while others start work on newer versions that are still in development. Translation speed in reduced by sharing resources across projects. Automatic suggestions, from a library of 750,000 translated strings, and a community of 19,000 registered translators can radically cut the time required to localise your project into many languages.

# 6 Summary

The communities we join, whether off-line or on-line, say a lot about the sort of people we are. The flip-side to this is that the tools you choose for your community - particularly the CDE and version control tool - can have a large impact on who joins and how easily they can contribute.

In their own right, Launchpad and Bazaar are highly useful tools. Together, they can:

- help your community track major assets such as source code and knowledge
- help it grow by reducing barriers to entry
- help it interact with related communities.

In particular, Launchpad is a free code hosting service for your Bazaar branches, branches can be browsed online, branches can be linked to bugs and blueprints, and the status of bug-branch relationships can be automatically managed by mentioning the bug while committing in Bazaar. Further integration is under development with the aim of streamlining the process from *great idea* to *running code in the hands of end users*.

If you have any feedback on how you'd like to see Bazaar and Launchpad further integrated, please contact us on the Bazaar mailing list, bazaar@lists.canonical.com.

While designed as a free service to support free software projects, Canonical may make Launchpad available to commercial software developers depending on their requirements. We would be happy to hear from you if you think Launchpad would be useful for managing your community.