Is your project registered yet?

Launchpad 1.0 Feature Highlights

30 March 2007

Table of Contents

Introduction....4
- Finding your way around....4
- What are you looking at?....5
- What application are you using?....5
- Are you logged in?....6
- What actions can you take here?....6
- What additional information is available?....6
- What help is available?....7
- Moving quickly to related items....7

Tracking bugs across multiple projects....8
- Bugs that affect multiple communities....8
- Several occurrences of the same bug in one project....9
- Fixing bugs in previous versions of software....10
- Bugs in external trackers....10

Monitoring bugs in other bug trackers....11
  - Automatically monitor the status of external bugs....11
  - Bug watches automatically notify Launchpad subscribers....11

Managing bugs through email....12
  - Email notification of new bugs in a project or package....12
  - Subscribing to specific bugs....12
  - Using teams to track bugs....12

Team management....13
- Subscription policies....13
- Team hierarchies....14
- Using teams....14

Bazaar branch hosting....15
- Everybody has revision control....15
- Anybody can publish branches....15
  - Publishing branches with Bazaar and Launchpad....16
  - Mirroring external Bazaar branches with Launchpad....16
  - Lowering barriers to participation....16
  - Creating your branch....17
  - Making changes to your branch....18
- Publishing branches....18

Team branches....19
- Bazaar checkouts....19
- Setting up a team branch....19
- Combining branches and checkouts....20
- Branch statuses and links....20

Fixing bugs in dedicated branches....21

Series: major stable and development branches....22
  - Naming series....22
- Series and branches....23

Interoperability with CVS and SVN....24
- Branch imports....24
- Import precision....25

Building a support network in the community....26
  - Tracking questions....26
- Answer contacts....26
- Follow-up through email....27
- Local language support....27
- The best answer....27
- An incremental knowledge base....28
- Escalation to developers....28

Translating your software.... .29
    Keeping track of progress.... 29
    Contributing translations.... 30
    Separating translations for different versions.... 30
    Translation teams.... 31
    Levels of permission.... 32
Blueprints: lightweight specifications.... 33
        Blueprint status and priority.... 33
        Storing the detail of a Blueprint.... 34
    Linking Blueprints.... 34
    Release management.... 34
        Helping improve communication.... 34
Milestones.... 35
Thank you!.... 36

Introduction

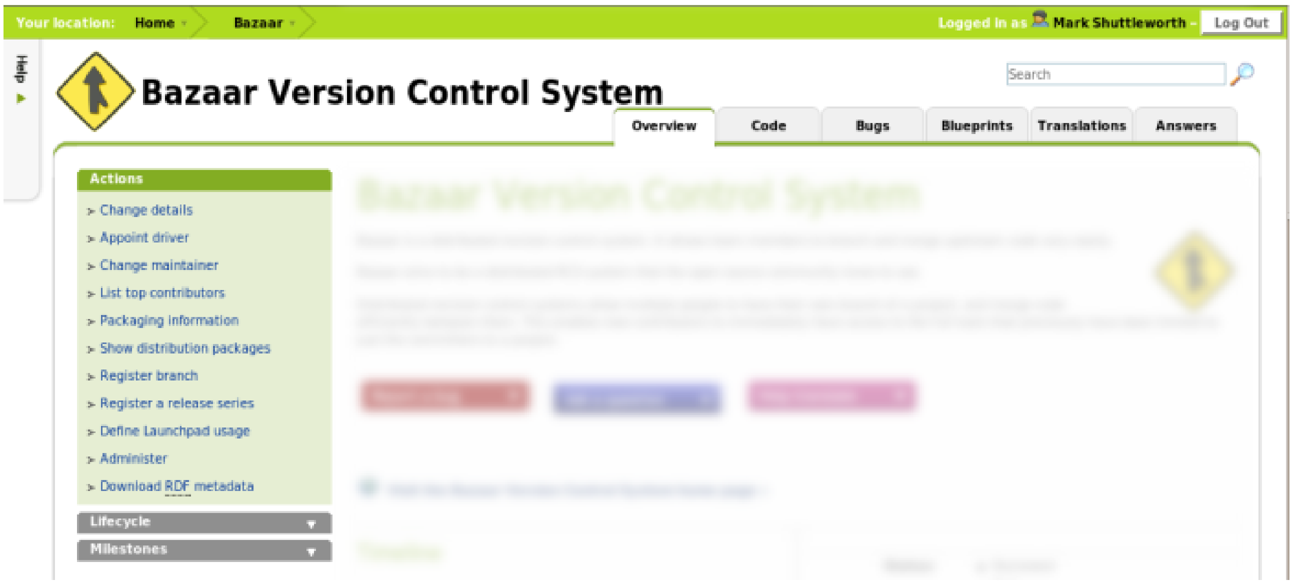
Launchpad is a tool that links communities together and makes them more productive. It gives you a window into the activity of each of those communities and allows you to connect with them and collaborate with them easily.

You can host your entire project on Launchpad, using it as a bug tracker, revision control repository and so on. Or you can just register your project in Launchpad, then link it to your existing hosting system and applications such as your bug tracker. Either way, Launchpad makes it easier to collaborate with other projects.

This Feature Guide will take you through a tour of some of the more interesting features in Launchpad. It's not a complete Tutorial or Documentation but it will give you a taste of Launchpad's functionality.

Finding your way around

Most pages in Launchpad look something like this:



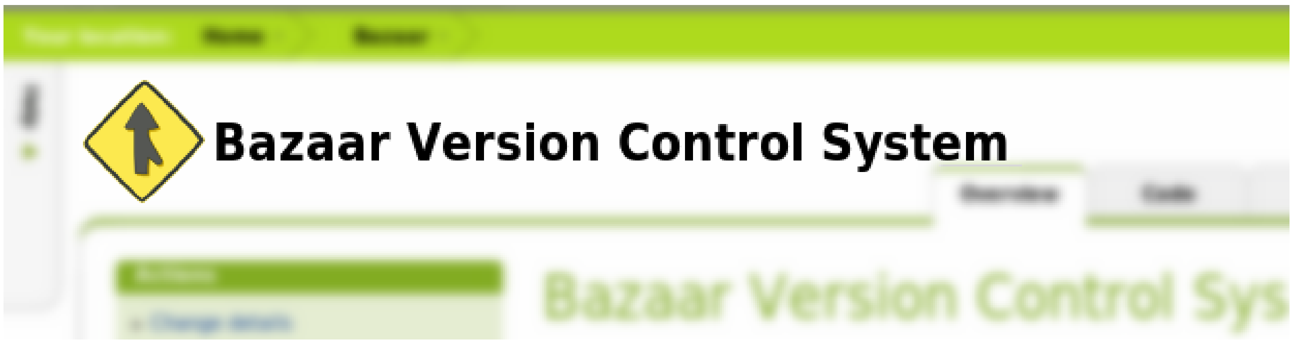
If you want to see this page in Launchpad itself, you'll find it at:

- https://launchpad.net/bzr

That tells you where you are in Launchpad, what you can do there, and what other applications might be relevant here. Let's highlight a couple of aspects of this page in brief.

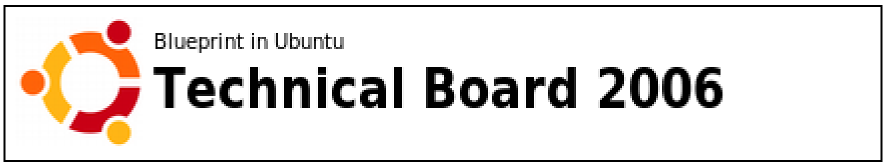
What are you looking at?

Right now, you are looking at a page about the Bazaar project. You know this is the case because the heading of the page has the Bazaar logo:



The graphic there can be customized for each project, and we recommend that you do so with any project that you register in Launchpad. Every page related to that project will carry the logo for that project (or a default one if none has been provided).

For example, here is the heading for an Ubuntu blueprint:



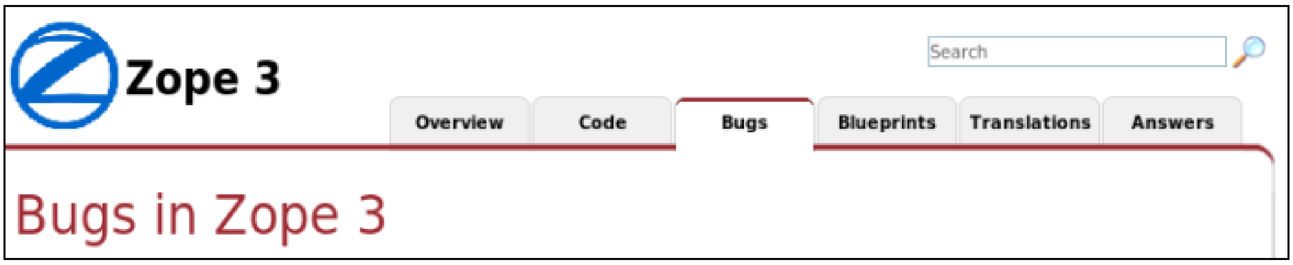
You can see that it is part of Ubuntu, and is a blueprint with the title "Technical Board 2006".

The heading will tell you which project, or source package, or blueprint you are looking at.

What application are you using?

Launchpad consists of several integrated applications, all of which understand free software projects, distributions, sprints, releases and milestones. You can tell which application you are looking at from the application tabs:

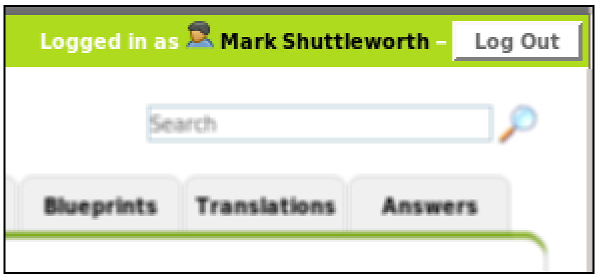
Here, for example, is a page dealing with bugs in Zope3:



You can also generally tell the application from the color of headings - we use the same colour throughout the system when dealing with a given application. In this case, bug-related pages will have red titles.

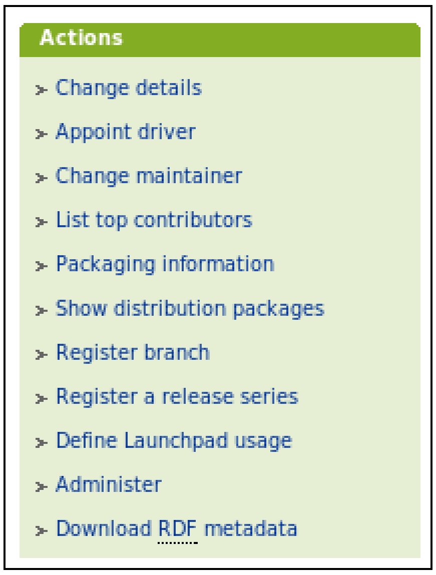
Are you logged in?

Some information or actions in Launchpad are only visible when you are logged in, and have the relevant permissions. You can easily tell if you are logged into Launchpad by looking on the top-right of any page in the system:



What actions can you take here?

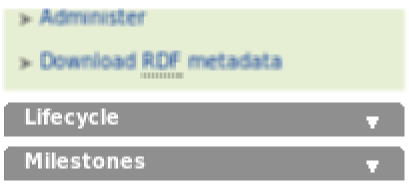
The "Actions menu" on most pages will give you the actions that are appropriate for this application, for this item.



The menu you see there is specific to the application - it changes between the bug tracker, the revision control system, the translation engine and so on. In general, actions that affect the registration of your project will be in the green ("Register" tab). For example, adding a new major version ("series") or a new milestone will happen there.

What additional information is available?

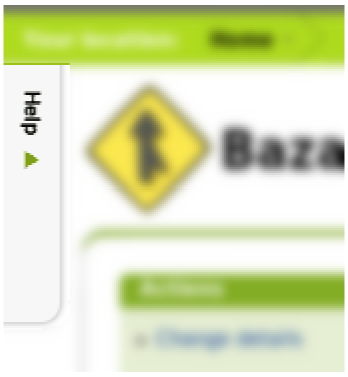
Sometimes, there are lists of additional information that are relevant but not essential. These will usually be displayed as expandable "portlets". For example:



Click on the title of the portlet to expand it. Some of these portlets are common to many items in the system. For example, a "Lifecycle" portlet will generally tell you when something was registered in Launchpad, and who did that.

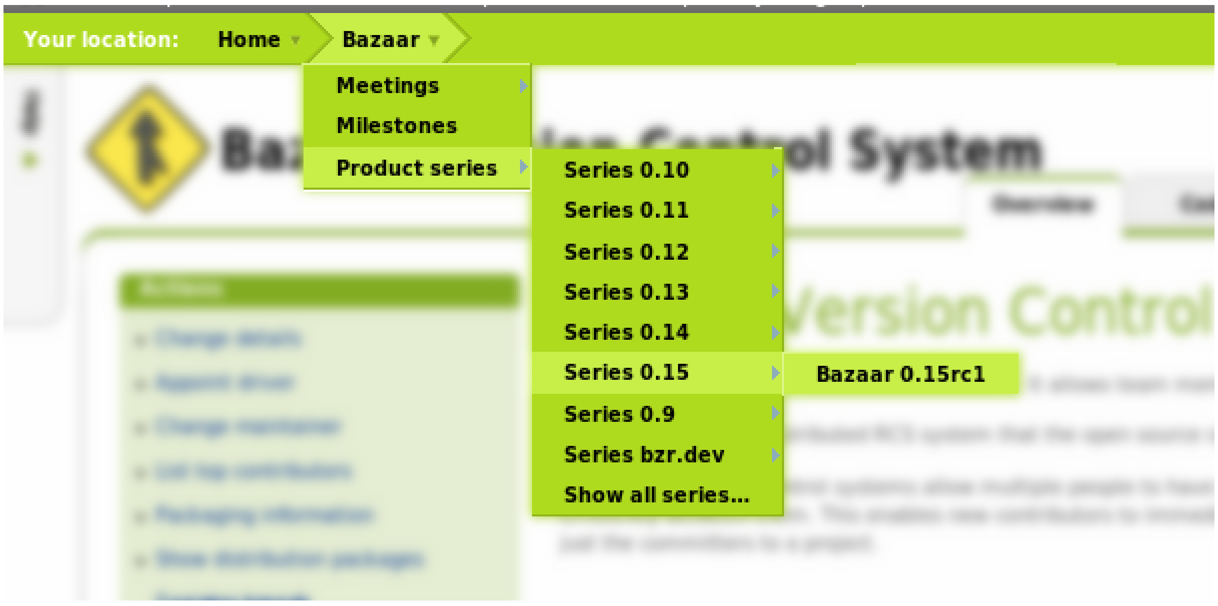
What help is available?

On every page you will see a small "Help" tab in the top left. Click that for additional information about the page or the action you are undertaking.



Moving quickly to related items

When you are looking at a page you may want to move quickly to a related item. For example, if you are looking at a page about Ubuntu 7.04 you may want to move quickly to a similar page for Ubuntu 6.06. The location and menu bar allows you to do just this. Here's a picture of the menu in use:



Now you know enough to explore the Launchpad applications. Let's start with one of the most unique features of Launchpad - tracking bug conversations and work across multiple communities.

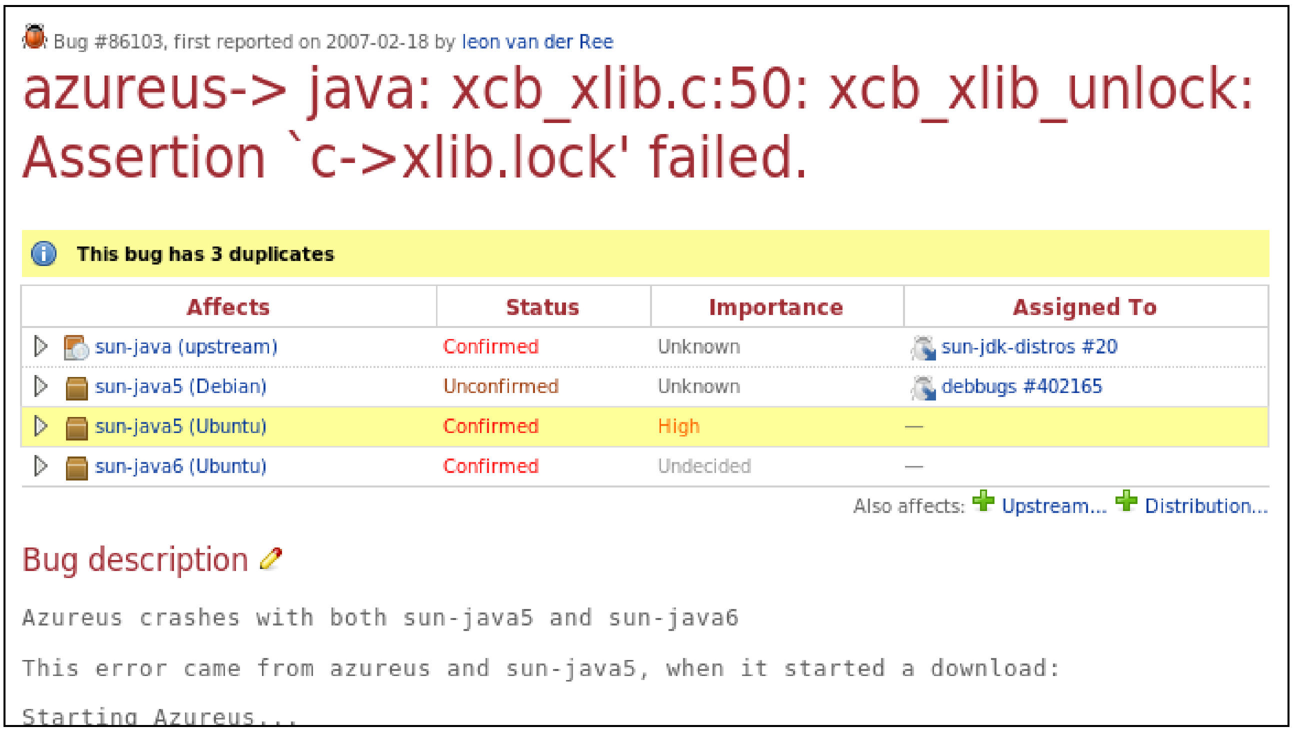
Tracking bugs across multiple projects

Launchpad's bug tracker is unique in that it allows you to track the status of **the same bug as it affects multiple different communities**. This gives you a one-page overview of the work that is ongoing in all of those communities to engineer a fix to the problem.

Bugs that affect multiple communities

Here's an example:

- https://launchpad.net/bugs/86103



Bug #86103, first reported on 2007-02-18 by leon van der Ree

azureus-> java: xcb_xlib.c:50: xcb_xlib_unlock: Assertion `c->xlib.lock' failed.

This bug has 3 duplicates

Affects	Status	Importance	Assigned To
sun-java (upstream)	Confirmed	Unknown	sun-jdk-distros #20
sun-java5 (Debian)	Unconfirmed	Unknown	debbugs #402165
sun-java5 (Ubuntu)	Confirmed	High	—
sun-java6 (Ubuntu)	Confirmed	Undecided	—

Also affects: Upstream... Distribution...

Bug description

```
Azureus crashes with both sun-java5 and sun-java6

This error came from azureus and sun-java5, when it started a download:

Starting Azureus...
```

The screen-shot shows that this bug currently affects at least three communities:

- Java upstream
- Debian
- Ubuntu.

In Ubuntu's case, the bug has been recorded in two places: the sun-java5 and sun-java6 packages.

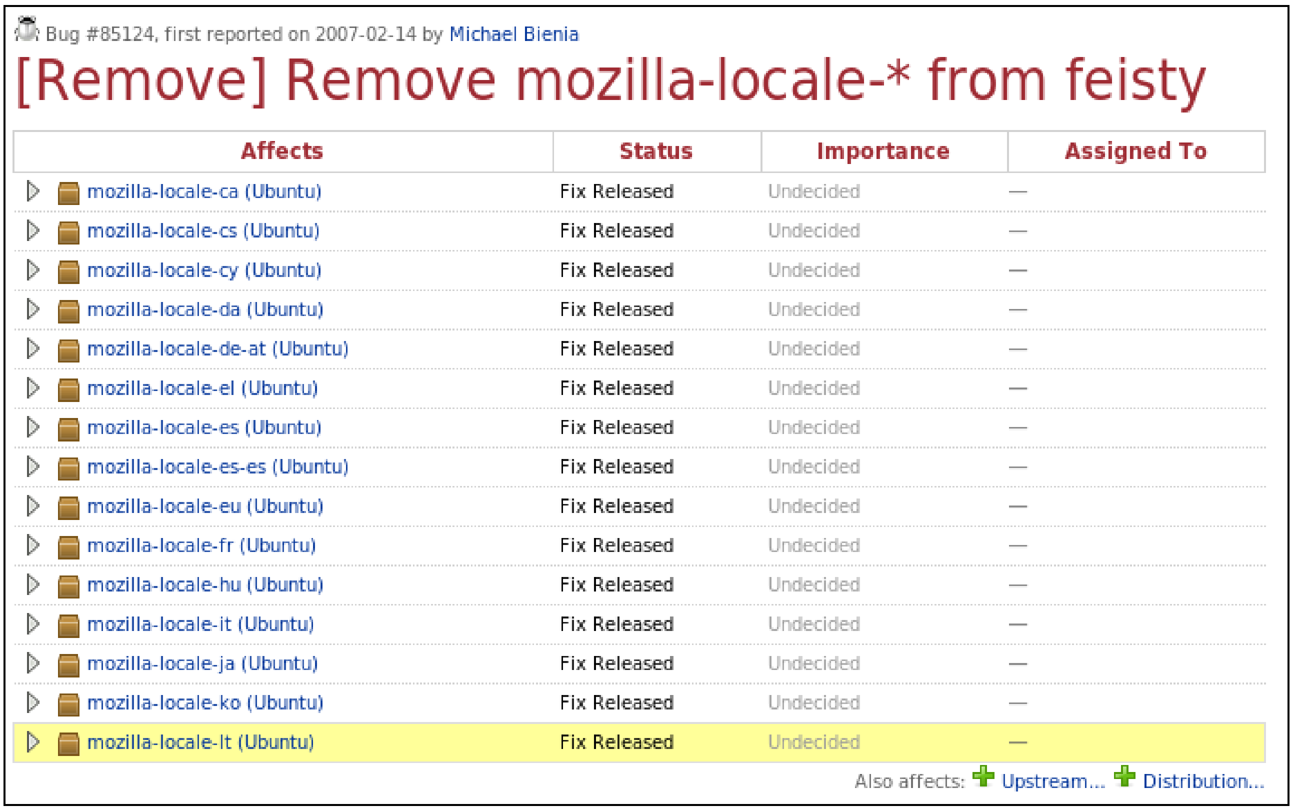
Each row in this table corresponds to a "place" where the bug has been detected, and so needs to be assessed and possibly fixed.

At the highest level, a "place" is a community, such as an upstream project or a distribution. There are also specialised types of place, which offer a finer level of granularity: for example, a major version of an application or a specific package within a distribution.

Several occurrences of the same bug in one project

It's possible for a bug to appear in many places in the same project. For example, this bug affects fifteen different packages in Ubuntu:

- https://launchpad.net/bugs/85124



Bug #85124, first reported on 2007-02-14 by Michael Bienia

[Remove] Remove mozilla-locale-* from feisty

Affects	Status	Importance	Assigned To
mozilla-locale-ca (Ubuntu)	Fix Released	Undecided	—
mozilla-locale-cs (Ubuntu)	Fix Released	Undecided	—
mozilla-locale-cy (Ubuntu)	Fix Released	Undecided	—
mozilla-locale-da (Ubuntu)	Fix Released	Undecided	—
mozilla-locale-de-at (Ubuntu)	Fix Released	Undecided	—
mozilla-locale-el (Ubuntu)	Fix Released	Undecided	—
mozilla-locale-es (Ubuntu)	Fix Released	Undecided	—
mozilla-locale-es-es (Ubuntu)	Fix Released	Undecided	—
mozilla-locale-eu (Ubuntu)	Fix Released	Undecided	—
mozilla-locale-fr (Ubuntu)	Fix Released	Undecided	—
mozilla-locale-hu (Ubuntu)	Fix Released	Undecided	—
mozilla-locale-it (Ubuntu)	Fix Released	Undecided	—
mozilla-locale-ja (Ubuntu)	Fix Released	Undecided	—
mozilla-locale-ko (Ubuntu)	Fix Released	Undecided	—
mozilla-locale-lt (Ubuntu)	Fix Released	Undecided	—

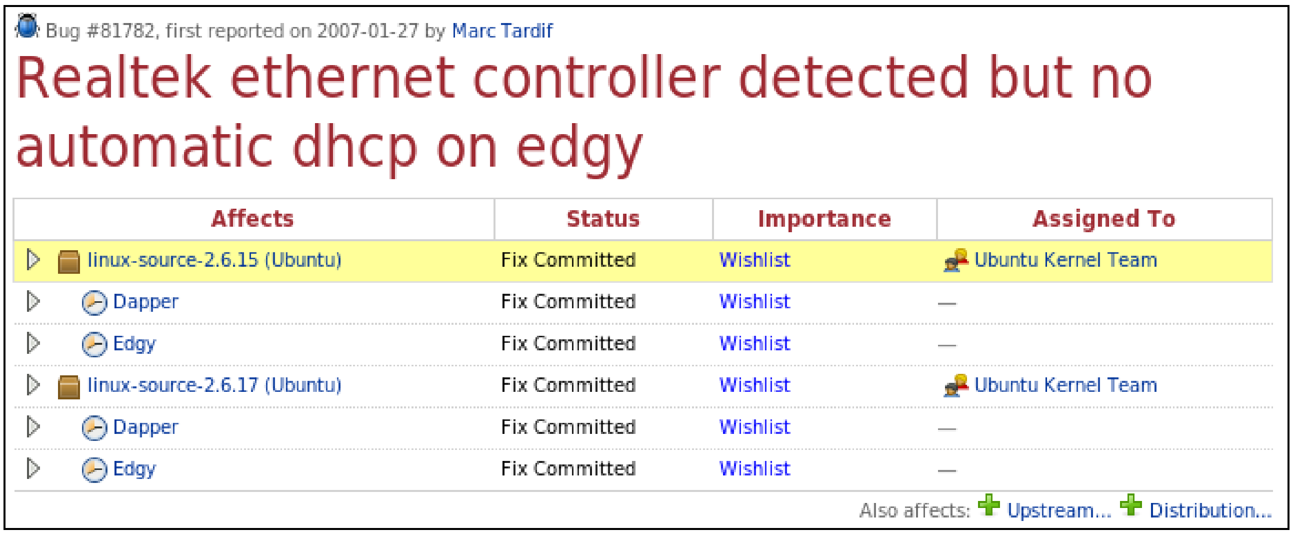
You can see that the issue has now been addressed in all packages. As a team works on a bug like that, they will fix different packages at different times. The bug page provides a useful overview of the real work still required before the bug can be considered to be fixed.

Fixing bugs in previous versions of software

Sometimes, a bug is severe enough that you need to fix it in previous versions of the software, perhaps as a security fix or an update to the stable and supported releases.

Here's an example of a bug that was deemed important enough to go back and fix in Dapper and Edgy (two previous releases of Ubuntu):

- https://launchpad.net/bugs/81782



Bug #81782, first reported on 2007-01-27 by Marc Tardif

Realtek ethernet controller detected but no automatic dhcp on edgy

Affects	Status	Importance	Assigned To
linux-source-2.6.15 (Ubuntu)	Fix Committed	Wishlist	Ubuntu Kernel Team
Dapper	Fix Committed	Wishlist	—
Edgy	Fix Committed	Wishlist	—
linux-source-2.6.17 (Ubuntu)	Fix Committed	Wishlist	Ubuntu Kernel Team
Dapper	Fix Committed	Wishlist	—
Edgy	Fix Committed	Wishlist	—

Also affects: Upstream... Distribution...

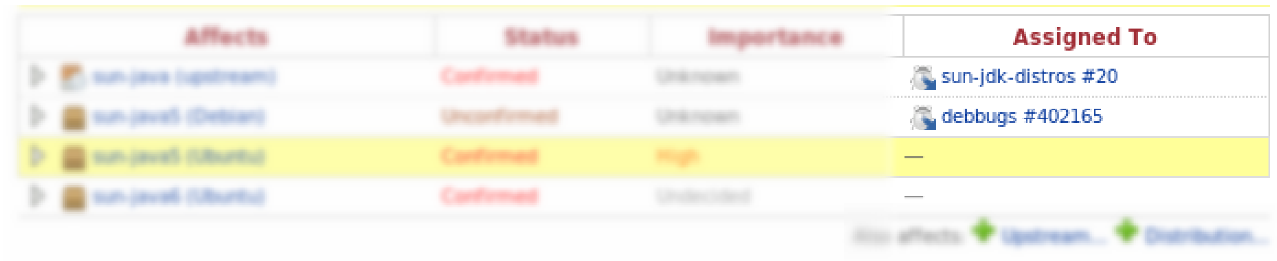
Note that this bug needed to be fixed in two packages: in the current development release and two stable releases of the distribution. This gives us a total of six rows in the table.

Bugs in external trackers

Now, look again at the table for bug #86103. You may notice that the icon in the assignee row looks unusual - it is not a "person" or a "team". That takes us to the next step in our tour - monitoring bugs in other bug trackers.

Monitoring bugs in other bug trackers

Look at the "Assigned To" column for bug #86103 again:



	Affects	Status	Importance	Assigned To
▷	sun-java (upstream)	Confirmed	Unknown	sun-jdk-distros #20
▷	sun-java5 (Debian)	Unconfirmed	Unknown	debbugs #402165
▷	sun-java5 (Ubuntu)	Confirmed	High	—
▷	sun-java6 (Ubuntu)	Confirmed	Undecided	—

Also affects: Upstream... Distribution...

The top two rows don't have a "person" assigned to fix them. Instead, because they refer to instances of the bug in external communities, they have a link to that bug as reported in the bug trackers used by those communities. Launchpad determines the external bug's status by regularly checking that external bug tracker.

Automatically monitor the status of external bugs

Open the bug in a new tab in your browser using this link, so you can interact with it more directly:

- https://launchpad.net/bugs/86103

Mouse over those two links in the "assignee" column and you will see URLs for the SUN Java and Debian bug trackers respectively:

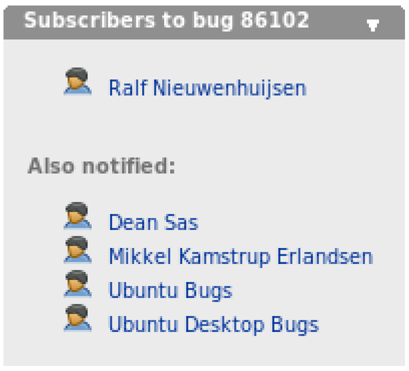
- https://jdk-distros.dev.java.net/issues/show_bug.cgi?id=20
- http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=402165

This is one of the unique features of Launchpad. It allows you to track the status of bugs in external trackers that run Bugzilla, RoundUp, Sourceforge or the Debian BTS. We will add support for additional trackers, when that becomes necessary to facilitate collaboration across the free software projects that use them.

This means that projects can leverage the benefits of Launchpad without giving up their own bug tracker, if they prefer to use their own hosted infrastructure. In some cases, we import ALL the bugs in a project bug tracker into Launchpad automatically, because there are other communities that want to be able to link to them trivially. In other cases, project members can create those links when they need them.

Bug watches automatically notify Launchpad subscribers

We call such a link a "Bug Watch" and you can create them for any bug in Launchpad. When you are telling Launchpad about a bug that affects another community, simply provide the external bug report's URL and Launchpad will automatically create the bug watch.



Subscribers to bug 86102

Ralf Nieuwenhuijsen

Also notified:

Dean Sas
Mikkel Kamstrup Erlandsen
Ubuntu Bugs
Ubuntu Desktop Bugs

Once created, Launchpad will monitor the remote bug report automatically and notify subscribers to the Launchpad bug of any changes to the status of the remote bug.

Incidentally, you can subscribe to any bug, and if you are the assignee or the bug contact for this package or project then you will also be treated as a subscriber, and hence also notified:

Now, scroll further down the page for Bug #86103. You may notice that some of the comments on the bug look like emails. That's because Launchpad's bug tracker allows you to interact with it completely via email. That's the next stop on our tour: the bug tracker email interface.

Managing bugs through email

If you prefer, you can entirely manage bugs using email.

Using the email interface you can:

- file a new bug
- comment on an existing bug
- assign a bug to someone
- change the status of a bug
- record that a bug affects a different community.

Email notification of new bugs in a project or package

You can ask Launchpad to email you every time someone reports a bug against a project or package that interests you, by registering as a bug contact for that project or package.

By replying to the new bug notification email, you can assign the bug to a Launchpad user or change its status, or just comment on the bug. For the details of how to use the email interface, please read our guide to UsingMaloneEmail.

Subscribing to specific bugs

If you are particularly interested in a specific bug, you can subscribe to it. Launchpad will email you every time someone comments on the bug or changes its status. If you reply to any of these emails, your comments will be sent to all of the other bug subscribers, and presented on the web site too. This is the best way to keep up to date with the progress of a bug.

Of course email notifications also work with remote bug watches. When you create a bug watch linking the Launchpad bug to a report in another community's bug tracker, Launchpad will notify all subscribers when there is a change to the remote bug's status. This is an excellent way to keep track of the big picture of community work on a bug.

Using teams to track bugs

Sometimes, a group of people may want to receive email notifications for a set of packages, or projects. With Launchpad, you can create a team and either subscribe that team to the bug or make the team a bug contact.

Launchpad will email everyone in that team every time a new bug is filed against the project or package.

You can find out more about Launchpad teams in the next step on our tour, Understanding Launchpad teams.

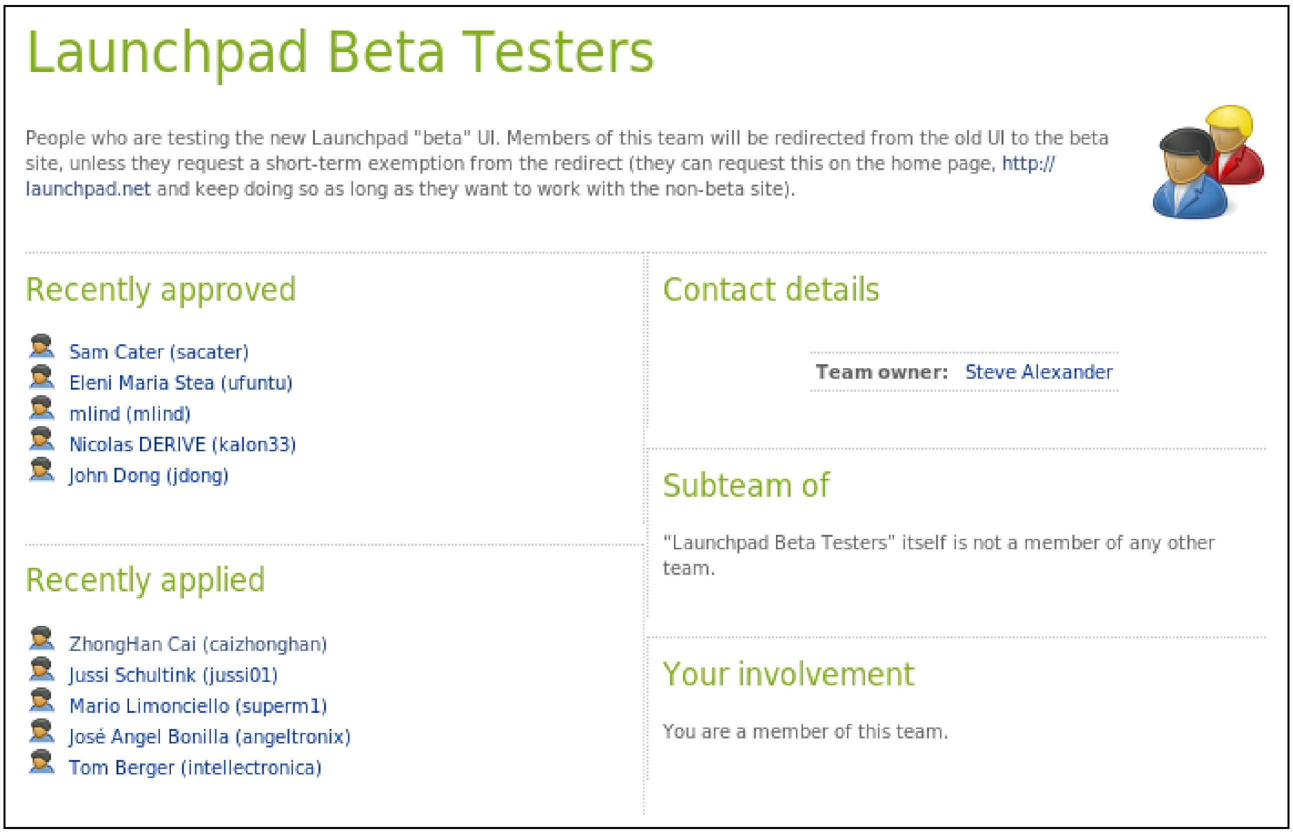
Team management

Strong communities are built on teams. Launchpad has a very rich infrastructure to support the creation of teams, which allows communities to develop in a sophisticated fashion around projects or initiatives.

Let's look at a team:

- https://launchpad.net/~launchpad-beta-testers/

Here's a snapshot of part of the page at the time of writing:



Launchpad Beta Testers

People who are testing the new Launchpad "beta" UI. Members of this team will be redirected from the old UI to the beta site, unless they request a short-term exemption from the redirect (they can request this on the home page, http://launchpad.net and keep doing so as long as they want to work with the non-beta site).

Recently approved

- Sam Cater (sacater)
- Eleni Maria Stea (ufuntu)
- mlind (mlind)
- Nicolas DERIVE (kalon33)
- John Dong (jdong)

Recently applied

- ZhongHan Cai (caizhonghan)
- Jussi Schultink (jussi01)
- Mario Limonciello (superm1)
- José Angel Bonilla (angeltronix)
- Tom Berger (intellectronica)

Contact details

Team owner: Steve Alexander

Subteam of

"Launchpad Beta Testers" itself is not a member of any other team.

Your involvement

You are a member of this team.

Subscription policies

As you can see, this has two lists of people: folks whose membership has been recently approved and people who have recently applied but have not yet been approved. This highlights a useful feature of Launchpad team management: membership subscription policies.

Launchpad allows you to choose from three types of membership subscription policy:

- **Entirely open:** anybody can join simply by adding themselves.
- **Moderated:** anybody can apply to join, but the team administrators must approve their membership. This is how the Launchpad Beta Testers team runs.
- **Restricted:** the only way to join a team is to be added by one of its administrators.

These subscription policies are similar to those used for mailing lists. They allow you, as the creator or administrator of a team, to decide how flexible you want to be, and to maintain good control of the growth of your community, if you choose.

Team hierarchies

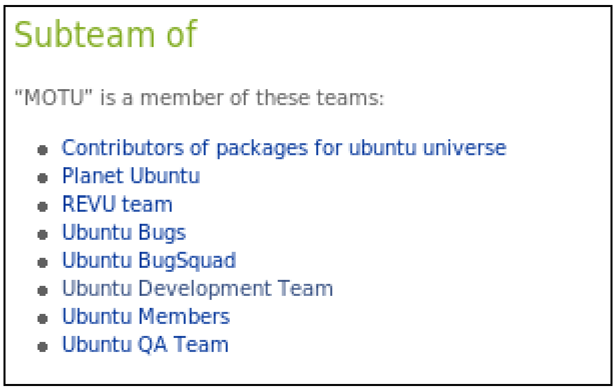
Notice that Launchpad Beta Testers is not a sub-team of any other teams? Launchpad allows you to make a team a member of another team.

For example: if Team A becomes a member of Team B, all the members of Team A effectively become members of Team B; that is, they get all the same permissions and access to all the same goodies. Technically they are "indirect" members of Team B, but that makes no difference, they are treated exactly the same as people who joined Team B directly.

Let's look at another team:

- https://launchpad.net/~motu/

Here is the relevant part of the screen:

- 

  Subteam of

  "MOTU" is a member of these teams:

  - Contributors of packages for ubuntu universe
  - Planet Ubuntu
  - REVU team
  - Ubuntu Bugs
  - Ubuntu BugSquad
  - Ubuntu Development Team
  - Ubuntu Members
  - Ubuntu QA Team

MOTU ("Masters of the Universe") is a very important team in Ubuntu because it's the one that looks after the largest collection of packages. It is also where most developers first get recognized as full contributors to Ubuntu.

As a result, the MOTU team is a member of several other teams. Structuring things that way means that "all new Ubuntu developers" become members of those other teams too.

There is no practical limit to the depth of team nesting that you can arrange. For example, we have some community teams that are organised by regions of a country. That country then also has a national team, consisting only of the regional teams. This way, everyone who is a member of a regional team is also a member of the national team but administration of individual memberships is handled by the regions.

Using teams

Creating a team is trivial. Once you have created it, you can add both people and other teams as members. You can then use your team just about anywhere that you might use an individual person. For example, you can make a team the assignee of a bug, or the driver of a project, or the translator of a piece of software.

One of the nicest uses of teams, however, is to determine who gets "commit access" to a particular branch of code. That leads us to the next stop on our tour- hosting branches of code in Launchpad.

Bazaar branch hosting

Launchpad can host branches of your project's code, using a revision control system called Bazaar. Since Bazaar is entirely distributed, it also allows you to register branches that are actually hosted elsewhere. As you've seen in our first examples, the project that develops Bazaar is itself hosted on Launchpad.

Regardless of where your code is hosted, Launchpad makes it accessible through a convenient catalog.

Bazaar is a very cool revision control system, powerful, flexible and fast. It's written in Python, so it's also easy to embed and modify. You can find out more on the Bazaar web site.

Everybody has revision control

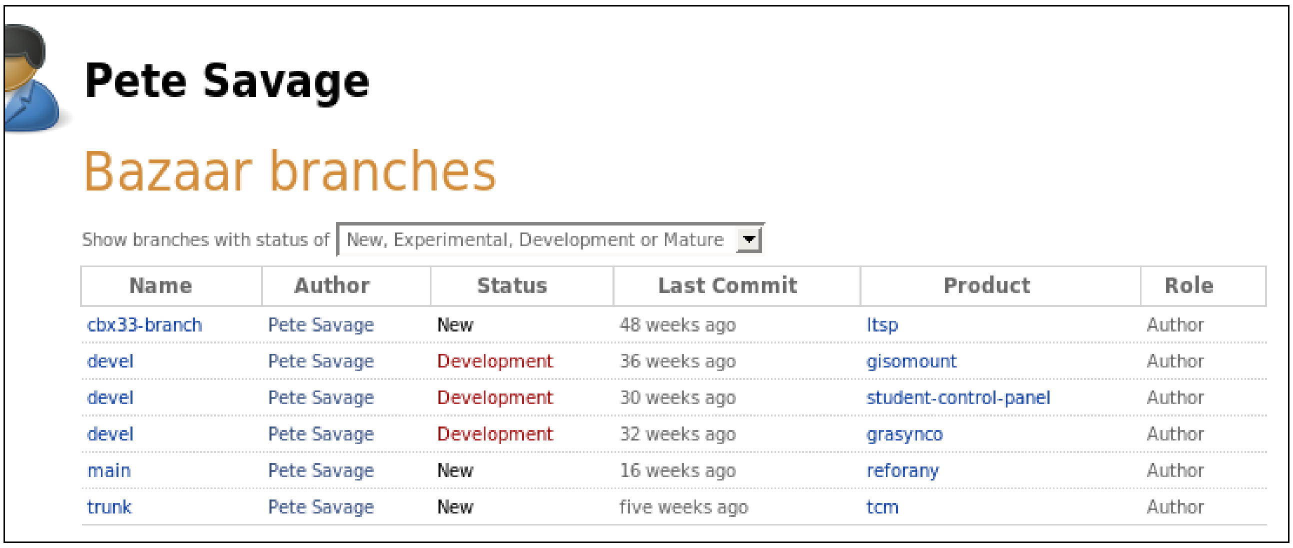
With CVS or SVN, there is a well-defined set of people who can make changes to the code that is centrally hosted in revision control. You can set up read-only access for other users, but they won't be able to keep track of their changes since they are not allowed to record those changes on the central server.

Bazaar eliminates that problem. Everybody has full revision control over their own version of the repository, even if they are not yet official committers to the project. Anybody can make their own branches, at any time, without having to ask permission first.

For example, let's look at the work of Pete Savage:

- https://code.launchpad.net/~petesavage

Here's the part of the page we're interested in, as it looked at the time of writing:



Pete Savage

Bazaar branches

Show branches with status of New, Experimental, Development or Mature

Name	Author	Status	Last Commit	Product	Role
cbx33-branch	Pete Savage	New	48 weeks ago	ltsp	Author
devel	Pete Savage	Development	36 weeks ago	gisomount	Author
devel	Pete Savage	Development	30 weeks ago	student-control-panel	Author
devel	Pete Savage	Development	32 weeks ago	grasynco	Author
main	Pete Savage	New	16 weeks ago	reforany	Author
trunk	Pete Savage	New	five weeks ago	tcm	Author

As you can see, Pete has been pretty busy! He's published branches of at least six projects in Launchpad: ltsp, gisomount, student-control-panel, grasynco, reforany and tcm.

Anybody can publish branches

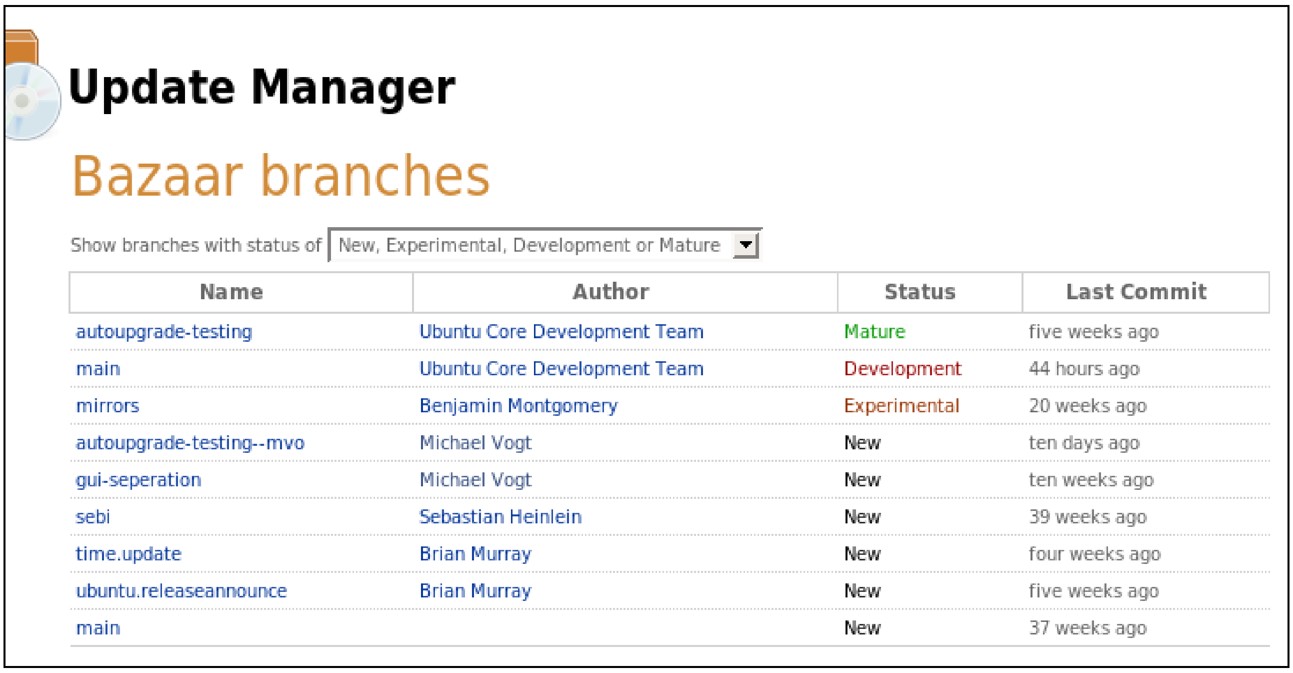
Not only can you create your own branches of ANY project in Launchpad, you can also publish those branches and have them show up in the listings of work in progress for the project. Importantly, you don't need any special permission to have your work show up in the listings for any project in Launchpad.

Major project branches - representing a stable line of development, or the trunk - do get special placement on those listings, but all the branches, including yours, will show up in the same place.

For example, take a look at the branches for Ubuntu's "Update Manager":

- https://code.launchpad.net/update-manager

The relevant part of the page at the time of writing is included here:



Update Manager

Bazaar branches

Show branches with status of: New, Experimental, Development or Mature

Name	Author	Status	Last Commit
autoupgrade-testing	Ubuntu Core Development Team	Mature	five weeks ago
main	Ubuntu Core Development Team	Development	44 hours ago
mirrors	Benjamin Montgomery	Experimental	20 weeks ago
autoupgrade-testing--mvo	Michael Vogt	New	ten days ago
gui-seperation	Michael Vogt	New	ten weeks ago
sebi	Sebastian Heinlein	New	39 weeks ago
time.update	Brian Murray	New	four weeks ago
ubuntu.releaseannounce	Brian Murray	New	five weeks ago
main		New	37 weeks ago

You can see that at least four people have their own branches of Update Manager: Benjamin Montgomery, Michael Vogt, Sebastian Heinlein, and Brian Murray. In addition, there are two branches published by the "Ubuntu Core Development Team". We'll take a closer look at team branches shortly.

Publishing branches with Bazaar and Launchpad

You can publish a branch on any server that you can write to, using FTP, SFTP, HTTP-Webdav or just by mounting the filesystem via NFS or another network protocol. Once it's published, you can share the branch with the world using plain HTTP or HTTPS. For faster performance you can use SFTP or the custom BZR protocol, which is designed to allow very fast access to branches - a very active area of development in Bazaar right now!

Launchpad includes an SFTP server to which you can push your branches. You can push branches for existing projects, or you can publish "any old stuff", which is unclassified.

Mirroring external Bazaar branches with Launchpad

If you have a branch published on your own server, you can still register it with Launchpad. Launchpad will monitor it and let interested people know whenever you add new code.

Launchpad will also make a backup of your code, which is accessible to everybody even if your server goes offline. And of course it means that your branch will show up in the project's listings.

Lowering barriers to participation

All of this is specifically designed to lower the barriers to participation in your project. A new developer who wants to make a small contribution can do so without having to first get permission to change the development trunk. He can branch from the trunk, work on his feature with full revision control, publish that branch to Launchpad or any other web hosting service, and eventually ask for his work to be merged into the trunk by a developer who does have commit rights there.

Let's take a closer look at the process of creating a branch.

Creating branches

For this section of the reviewers guide, you might like to actually create and publish a branch or two. You'll need to have Bzr installed and to have registered SSH keys with Launchpad, which you can use to authenticate yourself.

For Ubuntu users, simply type "sudo apt-get install bzr" and you will be all set. For users of other platforms, take a look at these instructions to install Bzr for yourself.

You should have at least version 0.15 of Bzr installed. To check:

```
% bzr version
Bazaar (bzr) 0.15.0candidate2
```

Alternatively, just read over the text to get a sense of what's possible.

Creating your branch

Now, let's create a branch of the famous "GNU Hello" application, which is used as a demonstration of several GNU best practices and technologies.

For simplicity we will use HTTP to fetch the code for this branch. This is quite an inefficient protocol, for this purpose, so it takes a little longer than normal to fetch the code. You can also use the optimised smart server protocol, once you have registered your SSH keys to access the Launchpad server securely.

We use HTTP here because it can be done anonymously.

-
  ```
  % bzr branch http://launchpad.net/gnuhello
  Branched 191 revision(s).
  ```

Done! You now have your own branch of GNU Hello.

-
  ```
  % cd gnuhello
  % ls
  ABOUT-NLS    ChangeLog     COPYING  Makefile.am  README        tests
  AUTHORS      ChangeLog.O   doc      man          README-alpha  THANKS
  autogen.sh   configure.ac  gnulib   NEWS         README.dev    TODO
  build-aux    contrib       INSTALL  po           src
  ```

You can see the latest commits on this branch. This is a branch of the trunk, so it has all the latest commits that trunk had when you created the branch.

-
  ```
  % bzr log | head --lines=13
  ------------------------------------------------------------
  revno: 191
  committer: karl
  timestamp: Tue 2007-02-13 23:09:30 +0000
  message:
    .
  ------------------------------------------------------------
  revno: 190
  committer: karl
  timestamp: Mon 2007-01-22 14:40:04 +0000
  message:
    update from texinfo
  ------------------------------------------------------------
  ```

Making changes to your branch

Because this is YOUR branch, you can commit to it immediately. Try making some changes, then type "bzr commit".

Before doing this, you can optionally configure Bazaar so that it knows who you are, and records that information with each commit. Type "`bzr whoami 'Joe Smith <email@domain.com>'`" to set that up.

Publishing branches

If you have set up some SSH keys in your Launchpad account, you can publish your branch with a single command.

You need to know the project name in Launchpad, your own Launchpad username, and of course the name you want to give this branch. If your branch is for personal use, and you don't want it to be listed in a specific project, you can call it "junk". Simply use `+junk` instead of the project name.

Continuing our example above, we will push this GNU Hello branch to Launchpad with the command:

- `% bzr push sftp://<me>@bazaar.launchpad.net/~<me>/gnuhello/mine`

Of course, you must substitute your Launchpad username for <me> in both places in the above command.

Now, if you take a look at your own branch listing page, you will see the GNU Hello branch listed:

- https://code.launchpad.net/people/+me (In this case leave the URL exactly as it appears. "+me" will translate to your actual account.)

As you can see, branching from an existing project, and publishing your branch, are extremely easy. Once your branch is published it is easy for others to find. If you would like your code to be included in the project's official line of development (and hence in the next release!) you can simply ask the project maintainers to review and merge your branch.

It may seem strange that you need your account name TWICE in the publishing command given above. The reason is that the first time (account@...) is to tell Launchpad who you are logging in as. The second time (~account/) is to tell it to put the branch into *your* directory.

This is needed because you can also publish branches into directories for each of the teams of which you are a member. And that leads us to the next stop on our tour - Team Branches!

Team branches

The combination of Bazaar branch hosting and teams gives you a very powerful capability to collaborate on code. Essentially, you can push a branch into a shared space, and anyone on that team can then commit to the branch.

This means that you can use Bazaar in the same way that you would use something like SVN, i.e. centrally hosting a branch that many people commit to.

Bazaar checkouts

It is possible for multiple people in a team each to "`bzr push`" their branches to the same location in the team space. For example, ~team/gnuhello/newfeature.

Bazaar will make sure that each push doesn't overwrite the work that is already there. Instead, it must extend that work. However, this is not usually the most optimal arrangement because each "push" can change the history of the branch in a dramatic way.

To get a more SVN-like experience, we usually recommend that people use Bazaar **checkouts** of a team branch. A checkout is essentially JUST the working code tree, without all the branch history, because the branch history stays on the central server.

When using Bazaar in this fashion it behaves very similarly to SVN. You cannot commit locally, because the knowledge of your branch history is on the remote server. But it does mean that you use less space locally, because you don't need to store all of that history locally too.

Setting up a team branch

To create a team branch, simply push a branch into a team space. For example, if you are still in the `gnuhello` branch you created during the earlier example, and you are a member of the `test-team` team, then you could create a shared branch of GNU Hello called "newfeature" using the following command:

-
  ```
  % bzr push sftp://<me>@bazaar.launchpad.net/~test-team/gnuhello/newfeature
  Created new branch.
  ```

Now, it is possible for anybody else to branch from that branch. It is also possible for anyone in the test-team to push an updated version of that branch to the same location. But the preferred approach, in general, is to encourage other team members to use a checkout of the branch:

-
  ```
  % bzr checkout sftp://<me>@bazaar.launchpad.net/~test-
  team/gnuhello/newfeature
  ```

Now, whenever they commit, Bazaar will first make sure they are up to date. If not, they can get up to date with `bzr update` and then commit.

Launchpad makes it extremely easy to administer the set of people who can commit to a branch like this, because they are simply the members of the team.

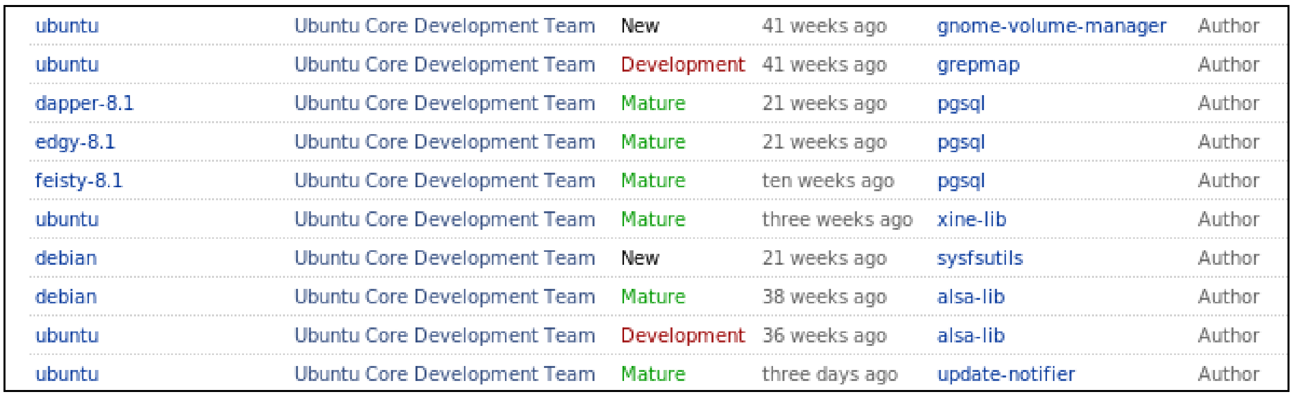
This means that it is trivial to create a team to collaborate on a feature. Create a new Launchpad team, with the people that you want to be able to commit to the feature's mainline branch. Push the initial branch to that team space. Then, tell everyone to commit there!

Team branches are a very popular way for the Ubuntu teams to collaborate. For example, you can set up teams around a single package or set of packages, and work on shared branches that contain the latest version of the relevant code.

The best example is the Ubuntu Core Development Team. It has branches for many projects that are shared and to which any team member can commit:

- https://code.launchpad.net/~ubuntu-core-dev

Here's a snippet from that page showing some of their branches:



ubuntu	Ubuntu Core Development Team	New	41 weeks ago	gnome-volume-manager	Author
ubuntu	Ubuntu Core Development Team	Development	41 weeks ago	grepmap	Author
dapper-8.1	Ubuntu Core Development Team	Mature	21 weeks ago	pgsql	Author
edgy-8.1	Ubuntu Core Development Team	Mature	21 weeks ago	pgsql	Author
feisty-8.1	Ubuntu Core Development Team	Mature	ten weeks ago	pgsql	Author
ubuntu	Ubuntu Core Development Team	Mature	three weeks ago	xine-lib	Author
debian	Ubuntu Core Development Team	New	21 weeks ago	sysfsutils	Author
debian	Ubuntu Core Development Team	Mature	38 weeks ago	alsa-lib	Author
ubuntu	Ubuntu Core Development Team	Development	36 weeks ago	alsa-lib	Author
ubuntu	Ubuntu Core Development Team	Mature	three days ago	update-notifier	Author

Notice how this is being used to keep track of packages that the team maintains both in Debian and Ubuntu. Shared branches can be used for cross-project collaboration in a very efficient way, with branches for specific projects and shared branches for work that is common to both of them.

Combining branches and checkouts

It is of course possible to get the best of both worlds, by combining branches and checkouts.

In the above example, a member of the team might have a checkout of the mainline branch to which they can commit, but then separately make their own branch locally which allows them to commit locally.

They would develop on their own local branch, perhaps pushing that up to the server in their own space rather than the team space. This gives them full revision control in their own branch. When they are ready to commit their work to the shared team mainline branch for the feature, they simply make sure their checkout of that branch is up to date, then merge from their local branch, and commit to the central server.

Branch statuses and links

The freedom to create branches is wonderful for encouraging participation. With all those branches out there, it's good to tell people which ones are most relevant to them. Pick good names for your branches! Also, use the branch status - New, Experimental, Mature, Obsolete, etc - to provide a hint to potential collaborators or testers about the maturity of your code.

One of the most useful things you can do is to link your branches to a description of the work they implement.

For example, if your branch fixes a bug, link the branch to the bug report! And if it's a new feature, track that feature in Launchpad (we call it a Blueprint) and link the branch to that.

That's the next stop in our review of Launchpad.

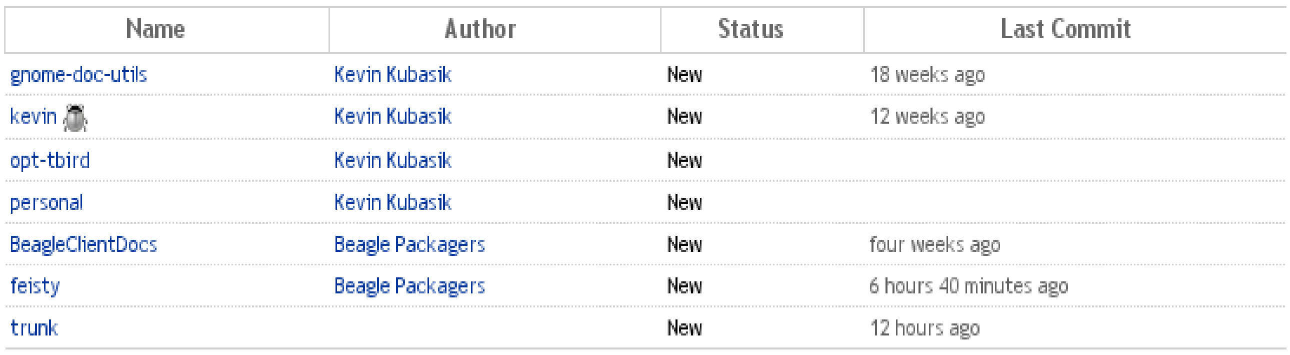
Fixing bugs in dedicated branches

If you create a Bazaar branch to fix a specific bug, and that branch is registered in Launchpad, you can link the bug report and the branch.

Launchpad uses an icon to indicate the link in the branch and bug listings. The icons are a quick way for anyone to see that you have code that is intended to fix a particular bug.

For example, take the branch listing for the Beagle project. You can see grey bug icons beside the name of some branches. Clicking on the bug icon takes you to the relevant page in the bug tracker.

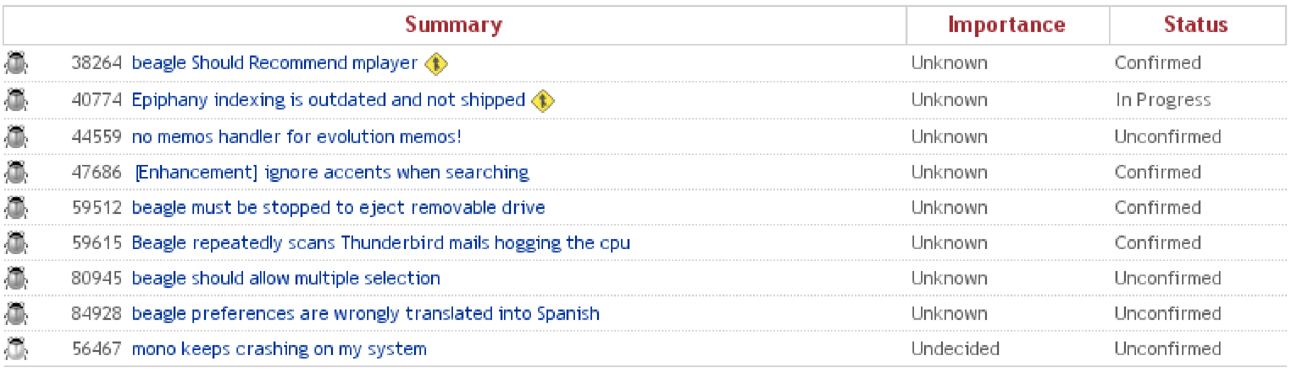
- http://code.beta.launchpad.net/beagle



Name	Author	Status	Last Commit
gnome-doc-utils	Kevin Kubasik	New	18 weeks ago
kevin	Kevin Kubasik	New	12 weeks ago
opt-tbird	Kevin Kubasik	New	
personal	Kevin Kubasik	New	
BeagleClientDocs	Beagle Packagers	New	four weeks ago
feisty	Beagle Packagers	New	6 hours 40 minutes ago
trunk		New	12 hours ago

Similarly, on the bug listing pages, a yellow Bazaar logo appears next to bugs that are linked to a branch. Clicking on the Bazaar icon takes you to the relevant branch.

- https://bugs.beta.launchpad.net/beagle/+bugs



Summary	Importance	Status
38264 beagle Should Recommend mplayer	Unknown	Confirmed
40774 Epiphany indexing is outdated and not shipped	Unknown	In Progress
44559 no memos handler for evolution memos!	Unknown	Unconfirmed
47686 [Enhancement] ignore accents when searching	Unknown	Confirmed
59512 beagle must be stopped to eject removable drive	Unknown	Confirmed
59615 Beagle repeatedly scans Thunderbird mails hogging the cpu	Unknown	Confirmed
80945 beagle should allow multiple selection	Unknown	Unconfirmed
84928 beagle preferences are wrongly translated into Spanish	Unknown	Unconfirmed
56467 mono keeps crashing on my system	Undecided	Unconfirmed

Each link between a bug and branch has its own status and whiteboard. The status indicates the progress of the fix, and you can use the whiteboard for more detailed information.

Any Launchpad user can have multiple branches of code for a project, and teams can have multiple branches too. How do you know which branch to use when starting new work?

For this, we have major branches in the project, which we call "Series", and that's what we will show you next.

Series: major stable and development branches

When we think of the way a project organises itself, there are some special branches: they represent major lines of development, and releases are "cut" from them. Typically these branches represent:

- **Development trunk:** this is the current "tip" of development across the core project community, representing the very cutting edge of work on that project. In general, the only releases made from the development trunk are snapshot, milestone or test releases, to generate more widespread testing and feedback.
- **Stable and supported branches:** these represent the latest work on the stable versions of the project, which are still supported. If updated stable releases are to be made they will come from these branches.
- **Obsolete branches:** for major release versions that are now out of date and no longer being updated. It's likely that work is no longer done on these branches.

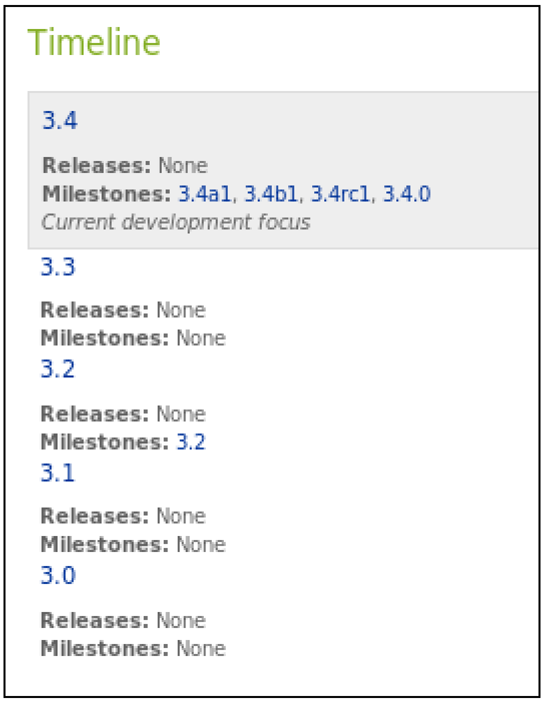
We call each of these "major" lines of development a "series", because they primarily represent a "series of releases of the same major version".

Note, there are usually multiple series for a given project, especially one that has produced stable releases. For example, a project might have a version 1.2 which is still supported, a 1.3 which is the "latest stable version" and recommended for new users, and a trunk which represents the current line of active development and is the source of snapshot test releases for the next stable version.

Take a look at the Zope3 project:

- https://launchpad.net/zope3

The stable and trunk series are shown in the "Timeline" section of the page:

- 

Timeline

3.4

Releases: None
Milestones: 3.4a1, 3.4b1, 3.4rc1, 3.4.0
Current development focus

3.3

Releases: None
Milestones: None

3.2

Releases: None
Milestones: 3.2

3.1

Releases: None
Milestones: None

3.0

Releases: None
Milestones: None

The trunk series here is identified with the words "Current development focus".

Naming series

There's some flexibility in the way projects name and organise themselves. For example, in the GNOME project, the "development trunk" is a series with an "odd" version number, such as 2.17. When it is ready for release, it is branched with an "even" version number, such as 2.18. Once GNOME make the stable release, the branch it to create 2.19, the new "trunk".

Sometimes, projects also call the trunk "MAIN", a term from the days of CVS. In the Zope3 example above, the 3.4 series was the trunk at the time of writing.

For convention, we encourage projects to call their development focus "trunk" and leave it as the same branch over time. When creating a new stable series, branch from trunk, create the series and link the branch to that series. This ensures that people who create a checkout of "trunk" don't find that it goes stale because it has been turned into a stable series, suddenly. And those who checkout or branch from a stable series will get what they expect, too.

Series and branches

In Launchpad, each project can register multiple series. Each of those series can be associated with a branch. By default, if you don't specify which series you are interested in, you will get the current development trunk series (**usually** called trunk, but not always). For example, in the example where you branched GNU Hello, you typed:

-
  ```
  % bzr branch http://launchpad.net/gnuhello
  191 revisions branched.
  ```

This command specifies only the project name, so you would have been given the development trunk. If there was a 1.3 series with a branch associated with it, you could have typed `bzr branch http://launchpad.net/gnuhello/1.3` and received the tip of that stable branch.

Note that this is different to the previous approach of branching from a specific person's version of a project. Here you are branching from one of the "major project branches", or series.

Not every series has a branch associated with it. Often, series are registered to handle a set of translations for a particular major version of the software, or so that bugs can be targeted there for release management. But we do encourage you to make sure, when you branch from trunk to make a new major stable release, that you set up a series for it and link the branch to it.

What if the project isn't yet using Bazaar officially, and code is being managed in CVS or Subversion? Launchpad can import the CVS or SVN trunk of the project and publish it for you as a Bazaar branch - and then keep it up to date so you can work entirely in Bazaar, merging from trunk whenever you want.

That's the next step in this tour.

Interoperability with CVS and SVN

Most free software projects still use CVS or Subversion (SVN) as their primary revision control system. They are well-understood and fast - for the core developers -systems. While they do have the disadvantage that branching and merging are not well supported, they're still effective and hence widely used.

The biggest downside to this approach, however, is that it treats newcomers and drive-by participants as second-class citizens. They don't get any form of revision control that can interoperate with that core team. Instead, they have to email patches, which can rot over time.

Launchpad and Bazaar can help you address this problem, while keeping the project trunk in the well-understood CVS or SVN.

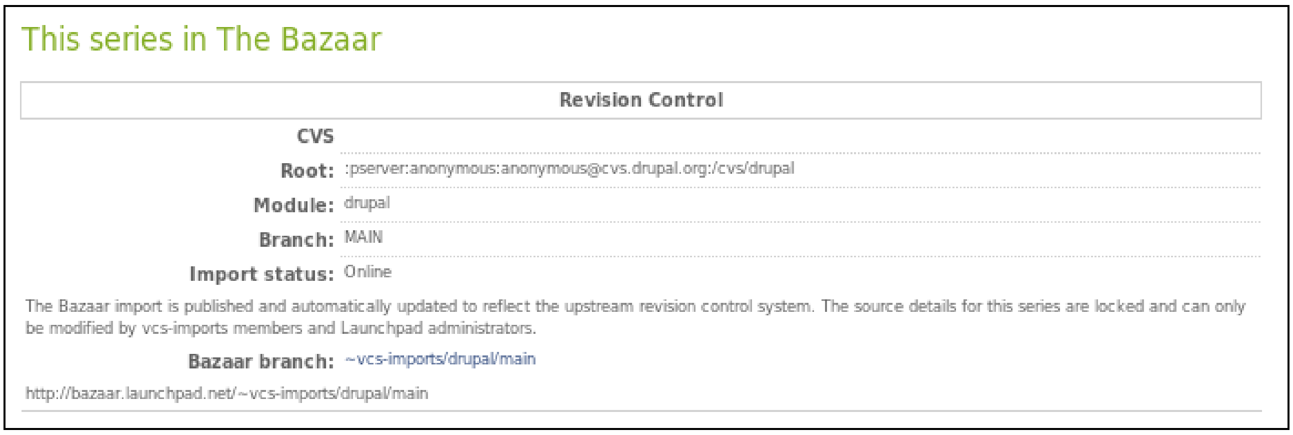
To do this, you need to set up a branch import.

Branch imports

These should more correctly be called "trunk imports" since Launchpad only supports an import of the primary development trunk of a project from CVS or SVN. A good example is the Drupal project, where there is an import of their trunk, recorded as the "main" Series of Drupal in Launchpad:

- https://launchpad.net/drupal/main

Here's a screenshot of the import details:



This series in The Bazaar

Revision Control	
CVS	
Root:	:pserver:anonymous:anonymous@cvs.drupal.org:/cvs/drupal
Module:	drupal
Branch:	MAIN
Import status:	Online

The Bazaar import is published and automatically updated to reflect the upstream revision control system. The source details for this series are locked and can only be modified by vcs-imports members and Launchpad administrators.

Bazaar branch: ~vcs-imports/drupal/main

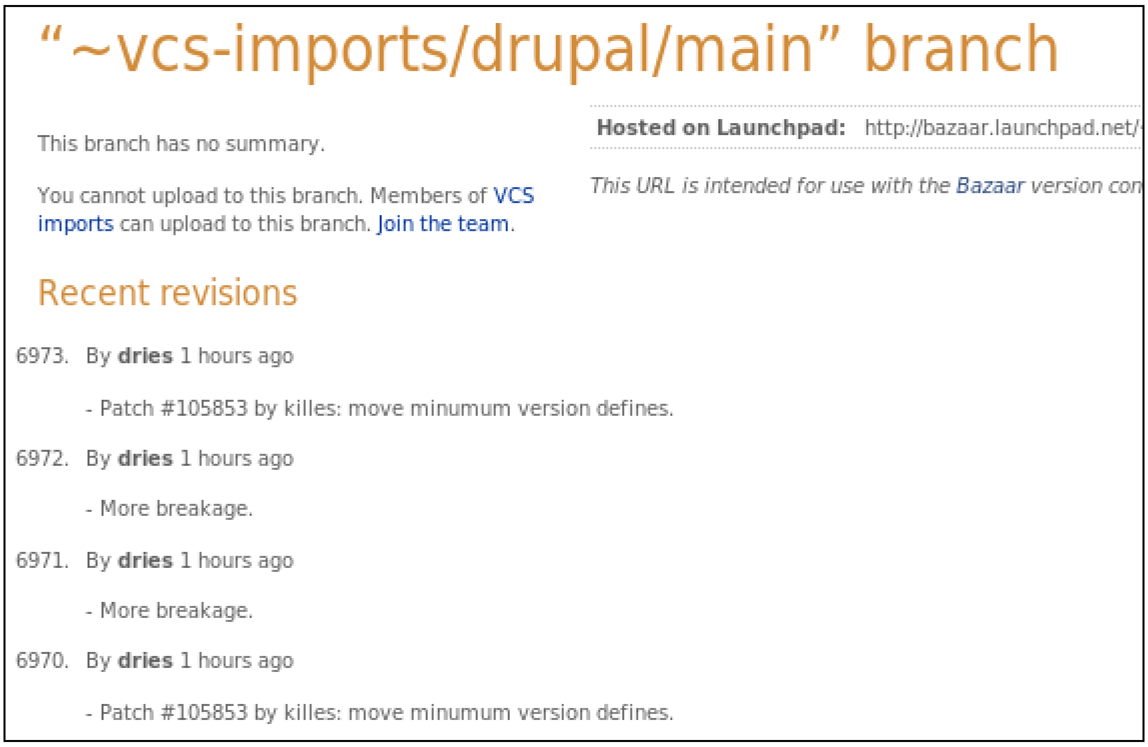
http://bazaar.launchpad.net/~vcs-imports/drupal/main

As you can see, the import is published as a Bazaar branch by the "vcs-imports" user. At the time of writing, the Drupal import was very active, with commits happening on the CVS trunk and being converted to Bazaar daily.

Here's the branch location, if you want to take a look at it now (or branch it, for that matter!):

- https://code.launchpad.net/~vcs-imports/drupal/main

And a snapshot of the latest commits at the time of writing:



“~vcs-imports/drupal/main” branch

This branch has no summary.

You cannot upload to this branch. Members of VCS imports can upload to this branch. Join the team.

Hosted on Launchpad: http://bazaar.launchpad.net/

This URL is intended for use with the Bazaar version con

Recent revisions

6973. By **dries** 1 hours ago

- Patch #105853 by killes: move minumum version defines.

6972. By **dries** 1 hours ago

- More breakage.

6971. By **dries** 1 hours ago

- More breakage.

6970. By **dries** 1 hours ago

- Patch #105853 by killes: move minumum version defines.

Import precision

Unfortunately, the initial import process is not an exact science. CVS and Subversion don't record enough information for a deterministic import into Bazaar, which is more rigorous about things like renames and changesets.

In most cases, where the CVS and SVN repositories have not been manually edited or altered, we can infer what we need and the import goes through smoothly. Sometimes, however, people have tried to work around limitations in CVS or SVN by altering the repositories behind-the-scenes. This is especially true of CVS, which does not support renames, so people have tended to do them manually.

So, a good import is part voodoo, part science, part luck. An import will not lose data - we can verify that the result of a checkout of the Bazaar branch is identical to a checkout of the CVS branch. But getting it to that point may well require inspection and custom work. For this reason, we don't have an automated process for the import. Instead, you request one, and we put it in a queue. Sometimes it takes just an hour or two, sometimes it can take days to get a good import together. In a very few cases, the old repositories are so wedged that we can't get all the history exactly right. Its best just to get started, and see how it goes. We are constantly improving the voodoo.

When you request an import, you use the "Answer Tracker", which is the next stop on our tour of Launchpad.

Building a support network in the community

In free software projects, the best "support" is often available directly from your own community. However, that support happens on informal channels - such as IRC, mailing lists and web forums. It can be difficult to capture those discussions, to turn them into a more structured knowledge-base.

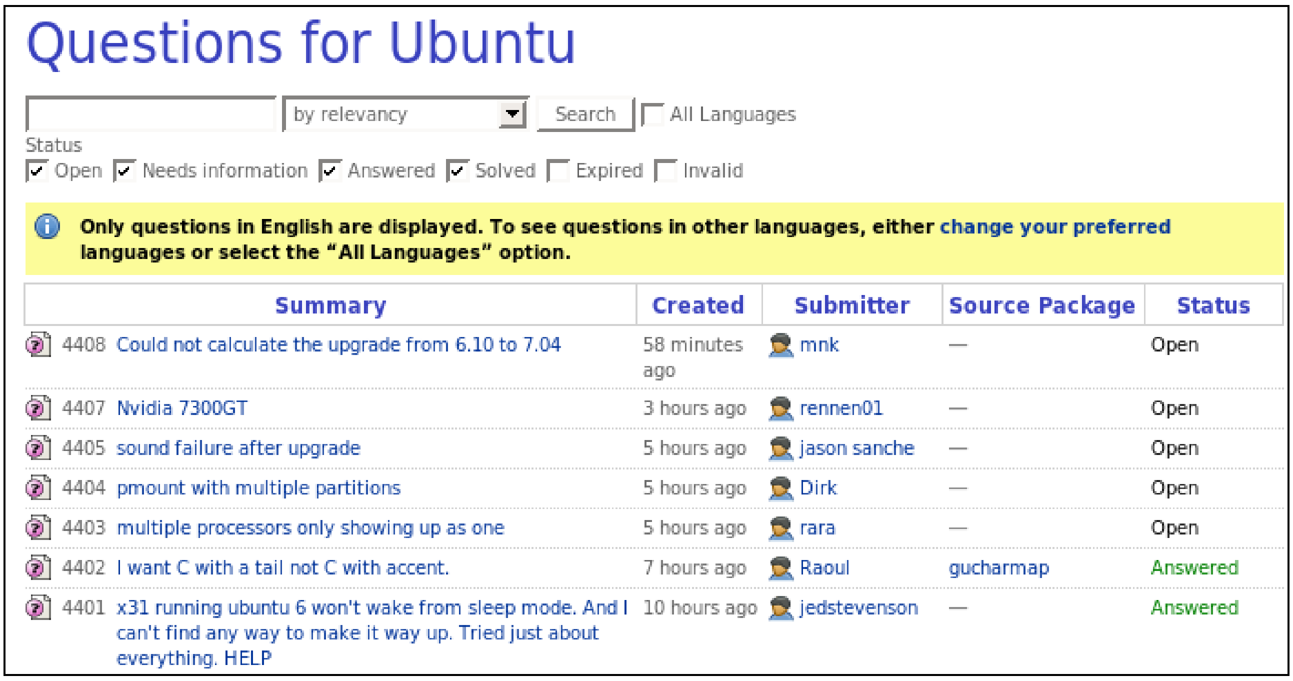
Launchpad's Answer Tracker provides a framework for just that. It's also useful as a way of tracking system administration requests - for example, to create new accounts in the project if you have shared systems for developers.

Tracking questions

Questions show up in lists for each project in the system. For example, here are the latest questions for Ubuntu:

- https://answers.launchpad.net/ubuntu

And here's a snapshot of those at the time of writing:



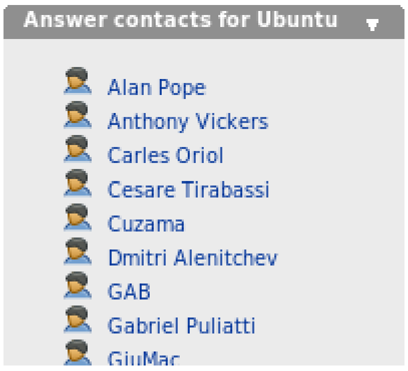
Questions for Ubuntu

Status
[✓ Open] [✓ Needs information] [✓ Answered] [✓ Solved] Expired Invalid

Only questions in English are displayed. To see questions in other languages, either change your preferred languages or select the "All Languages" option.

	Summary	Created	Submitter	Source Package	Status
4408	Could not calculate the upgrade from 6.10 to 7.04	58 minutes ago	mnk	—	Open
4407	Nvidia 7300GT	3 hours ago	rennen01	—	Open
4405	sound failure after upgrade	5 hours ago	jason sanche	—	Open
4404	pmount with multiple partitions	5 hours ago	Dirk	—	Open
4403	multiple processors only showing up as one	5 hours ago	rara	—	Open
4402	I want C with a tail not C with accent.	7 hours ago	Raoul	gucharmap	Answered
4401	x31 running ubuntu 6 won't wake from sleep mode. And I can't find any way to make it way up. Tried just about everything. HELP	10 hours ago	jedstevenson	—	Answered

Note that some of the questions are general "Ubuntu" questions, and others are focused on the use of a specific package.



Answer contacts for Ubuntu

- Alan Pope
- Anthony Vickers
- Carles Oriol
- Cesare Tirabassi
- Cuzama
- Dmitri Alenitchev
- GAB
- Gabriel Puliatti
- GiuMac

Answer contacts

There's no joy in posting questions into a vacuum. Launchpad lets members of your community become "answer contacts", who are notified each time a new question is asked. Here are some of the answer contacts for Ubuntu:

In the case of Ubuntu, you can become an answer contact for any package, or for the distribution as a whole. This lets people focus on the areas they know best, or choose to be generalists who try to help whatever issue a user has.

Follow-up through email

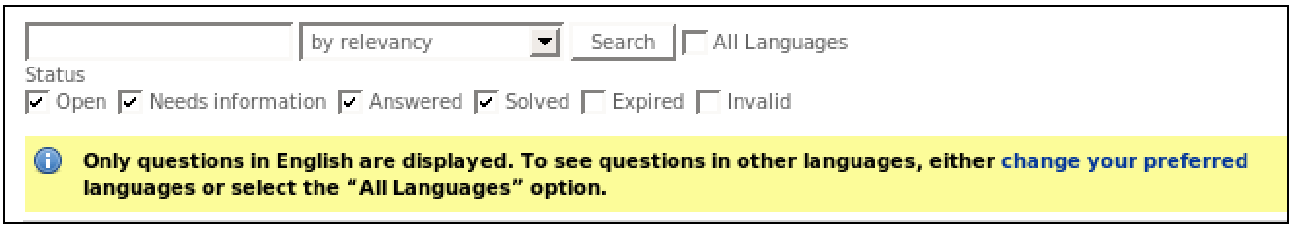
When answer contacts receive email notifying them of new questions, they can reply to the question directly from their email client. The conversation between the person asking the question, and those helping them, can happen entirely by email, with no need to move to the web interface. In fact, any interested user can subscribe to any question and participate in the discussion by email.

Local language support

Free software is a global phenomenon. By and large, developers of free software speak English, so much of the correspondence about bugs and features happens in that language.

When it comes to end-user support, however, local languages are key. Launchpad allows you to specify the languages that you speak. It then tries to connect people asking questions in a given language with people offering to answer those questions in the same language.

By default, you will also only see questions in the languages that you have set in your Launchpad user profile.



Status
[✓ Open] [✓ Needs information] [✓ Answered] [✓ Solved] Expired Invalid

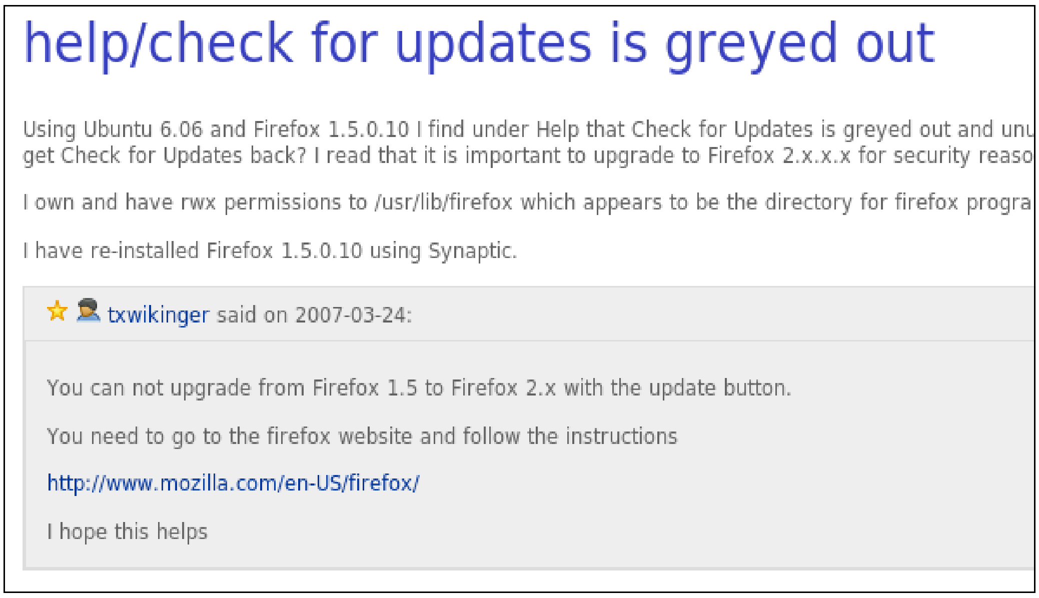
Only questions in English are displayed. To see questions in other languages, either change your preferred languages or select the "All Languages" option.

The Answer Tracker is one of the pieces of Launchpad that we intend to make entirely accessible in your local language, because it's designed to bring together communities in every corner of the globe to harness their shared knowledge and to help new users become comfortable with free software.

The best answer

Some questions take a few rounds of Q&A before the problem is diagnosed and the correct answer is given. For this reason, the user can specify which answer was the one that solved his problem. Launchpad then displays that answer more prominently so as to help the next user who has the same problem.

Here's an example of part of a page with the best answer highlighted:

- 

help/check for updates is greyed out

Using Ubuntu 6.06 and Firefox 1.5.0.10 I find under Help that Check for Updates is greyed out and unu
get Check for Updates back? I read that it is important to upgrade to Firefox 2.x.x.x for security reaso

I own and have rwx permissions to /usr/lib/firefox which appears to be the directory for firefox progra

I have re-installed Firefox 1.5.0.10 using Synaptic.

txwikinger said on 2007-03-24:

You can not upgrade from Firefox 1.5 to Firefox 2.x with the update button.

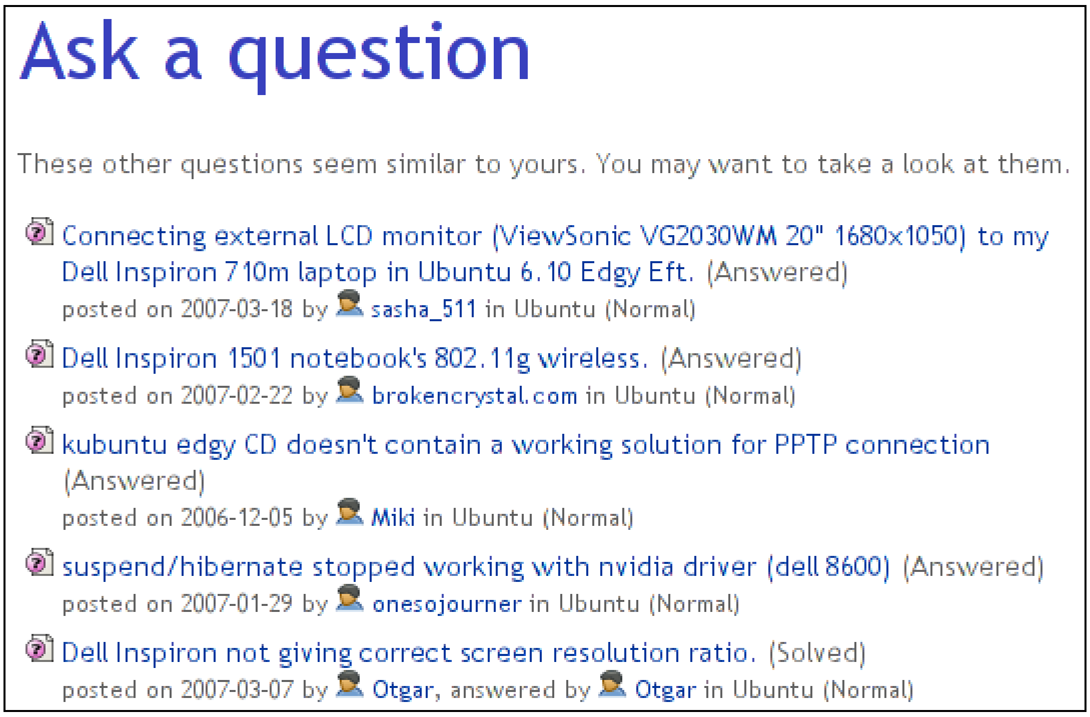
You need to go to the firefox website and follow the instructions

http://www.mozilla.com/en-US/firefox/

I hope this helps

An incremental knowledge base

When a user asks a question, they will see a list of similar questions that have previously been answered:



Ask a question

These other questions seem similar to yours. You may want to take a look at them.

- Connecting external LCD monitor (ViewSonic VG2030WM 20" 1680x1050) to my Dell Inspiron 710m laptop in Ubuntu 6.10 Edgy Eft. (Answered)
  posted on 2007-03-18 by sasha_511 in Ubuntu (Normal)
- Dell Inspiron 1501 notebook's 802.11g wireless. (Answered)
  posted on 2007-02-22 by brokencrystal.com in Ubuntu (Normal)
- kubuntu edgy CD doesn't contain a working solution for PPTP connection (Answered)
  posted on 2006-12-05 by Miki in Ubuntu (Normal)
- suspend/hibernate stopped working with nvidia driver (dell 8600) (Answered)
  posted on 2007-01-29 by onesojourner in Ubuntu (Normal)
- Dell Inspiron not giving correct screen resolution ratio. (Solved)
  posted on 2007-03-07 by Otgar, answered by Otgar in Ubuntu (Normal)

Each answered question builds the knowledge base for your project.

Escalation to developers

Sometimes, a user's problem is actually a bug in the software. It is easy for one of the answer contacts to create a bug report from the question, preserving the link back to the question and ensuring that the person who found the problem stays involved in the discussion.

Of course, at this point language can become a barrier, but in many cases the person who was trying to help answer the problem can act as a translator, facilitating communication between the developers and the person who discovered the issue, until the developers themselves can reproduce it.

In summary - Launchpad makes it straightforward to build a community team that is dedicated to helping solve end-user problems, and acting as a "first line of defence" in detecting problems that might be indicative of real bugs. Importantly, it specifically handles people asking questions in their language of choice.

And that leads us nicely to the next stop in our tour - Launchpad's software translation service, which we call "Rosetta".

Translating your software

Launchpad has unique infrastructure to support the translation of free software into many languages. It enables translation communities to form and organise themselves, leveraging the Team Management capabilities described earlier, and allows you to keep track of the translations of multiple versions of your software into multiple languages.

This translation infrastructure is all web-based, so new translators can join very easily, without any special tools or needing to commit to the project's codebase. However, Rosetta imports and exports standard translation files, enabling more advanced translators to use their preferred tools and avoid the web interface entirely.

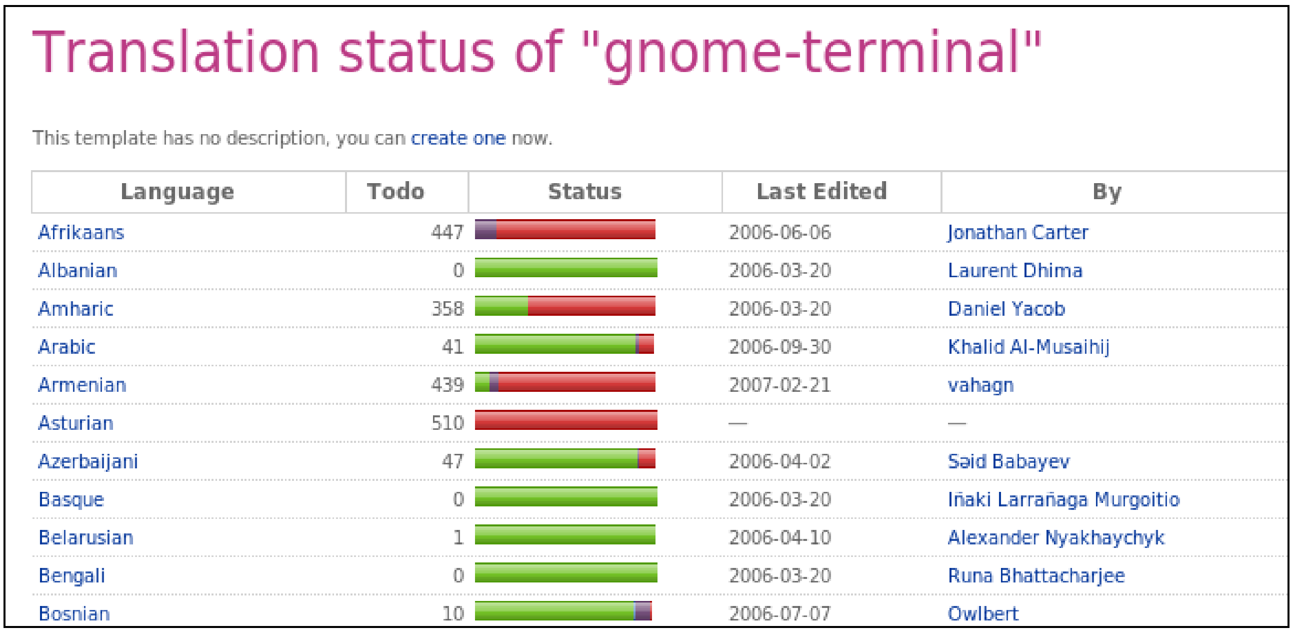
Ubuntu uses Launchpad to manage the translation of several hundred packages into many different languages, for each successive release. In particular, once a release has been made, Ubuntu translators can continue to improve translations, which are delivered to users in the form of updated language packs every few weeks.

Keeping track of progress

When software is ready to be translated, it results in one or more "translation templates". You can see the extent to which a particular template has been translated very easily. For example, here is the status of the Gnome Terminal translation in Ubuntu's 6.06 LTS release:

- https://translations.launchpad.net/ubuntu/dapper/+source/gnome-terminal/+pots/gnome-terminal

And part of that page, at the time of writing looked like this:



Translation status of "gnome-terminal"

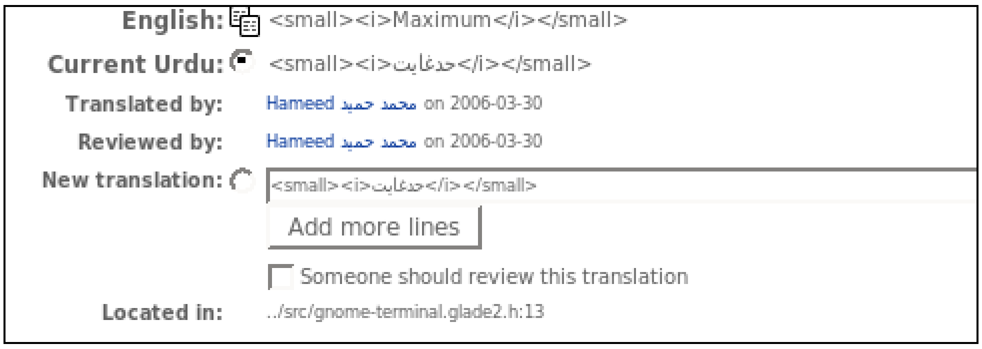
This template has no description, you can create one now.

Language	Todo	Status	Last Edited	By
Afrikaans	447	🟪🟥	2006-06-06	Jonathan Carter
Albanian	0	🟩	2006-03-20	Laurent Dhima
Amharic	358	🟩🟥	2006-03-20	Daniel Yacob
Arabic	41	🟩🟥	2006-09-30	Khalid Al-Musaihij
Armenian	439	🟩🟪🟥	2007-02-21	vahagn
Asturian	510	🟥	—	—
Azerbaijani	47	🟩🟥	2006-04-02	Said Babayev
Basque	0	🟩	2006-03-20	Iñaki Larrañaga Murgoitio
Belarusian	1	🟩	2006-04-10	Alexander Nyakhaychyk
Bengali	0	🟩	2006-03-20	Runa Bhattacharjee
Bosnian	10	🟩🟪	2006-07-07	Owlbert

You can immediately tell which languages are well-translated (green) and which are not. The purple bars represent new translations, added to this system since the package itself was built. It's also possible to see when someone last added to the translation of this template in any particular language, and who that person was.

Contributing translations

It's very easy for people to contribute new translations. Each translation is presented as a web form, with the English and current translation as well as suggestions based on other known translations prominently displayed. Here's an example of the form to translate a single message in an application (Gnome Terminal again):



English:	<small><i>Maximum</i></small>
Current Urdu:	<small><i>حدغایت</i></small>
Translated by:	Hameed محمد حمید on 2006-03-30
Reviewed by:	Hameed محمد حمید on 2006-03-30
New translation:	<small><i>حدغایت</i></small>
	Add more lines
	Someone should review this translation
Located in:	../src/gnome-terminal.glade2.h:13

Notice that the translator has included the relevant HTML display markup. The translation is substituted in the interface for the English string in its entirety. Some more sophisticated translations can contain multiple lines of text, and placeholders for values that come from the application itself. And of course, translations sometimes need to deal with plurals - the difference between "there are 9 cars on the lot" and "there is 1 car on the lot".

It takes time and skill to become a very good translator. For this reason, it is possible to have a core group of trusted translators who can modify any translation. Everyone else is limited to making suggestions, until they are trusted enough to be part of the core team.

Separating translations for different versions

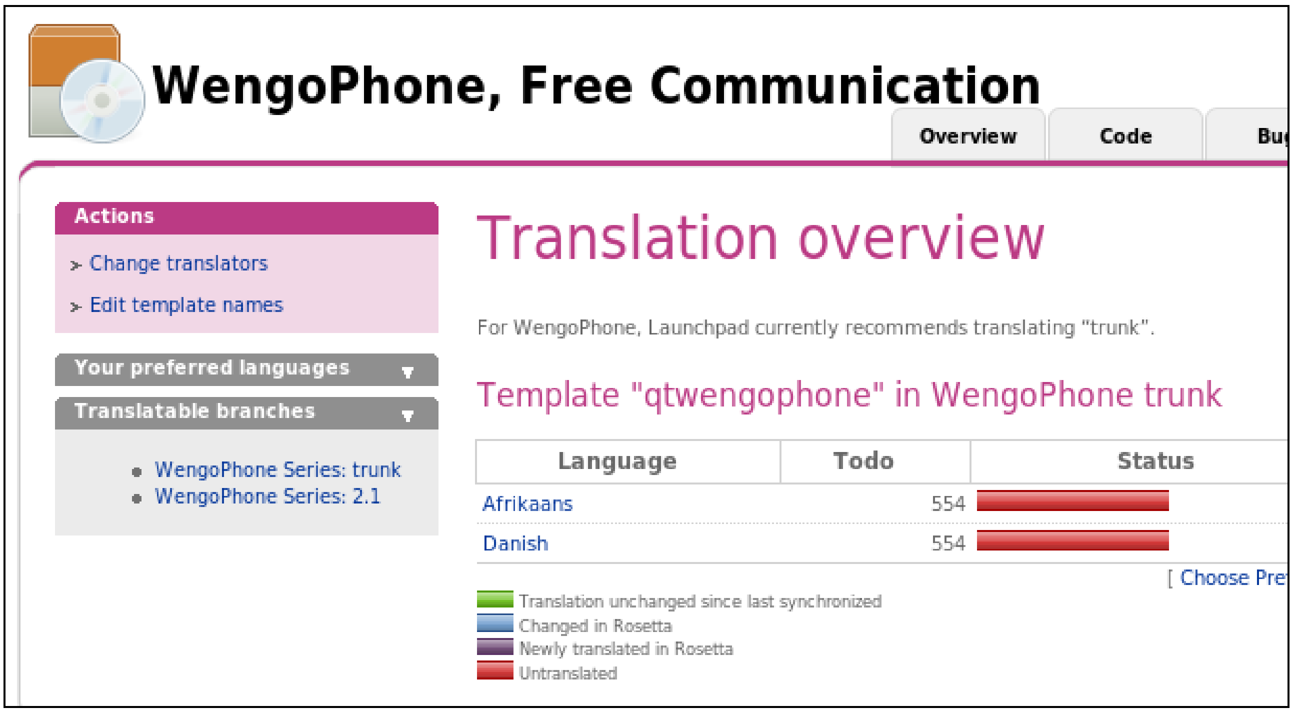
Often, you may want translators to be able to work on a stable version of the project code, and at the same time get a head start on the development version. Launchpad supports this explicitly. Each Series (we described those earlier - major versions of a project, such as 1.1.x, 1.2.x and trunk) can have its own set of translation files. When a translation is added to a Series, it is immediately suggested to the other if the original English string is unchanged.

This mean that some translators can focus on expanding the translation coverage of your stable release, while others work on the cutting edge.

For example, here you can see the work being done to translate WengoPhone. By default, the "trunk" is shown because the WengoPhone developers have identified that as the most important series to be translated, at the time of writing. But it is also possible to translate WengoPhone 2.1:

- https://translations.launchpad.net/wengophone/

And here's a snapshot of the relevant part of the page, at the time of writing:



WengoPhone, Free Communication

Actions

- Change translators
- Edit template names

Your preferred languages

Translatable branches

- WengoPhone Series: trunk
- WengoPhone Series: 2.1

Translation overview

For WengoPhone, Launchpad currently recommends translating "trunk".

Template "qtwengophone" in WengoPhone trunk

Language	Todo	Status
Afrikaans	554	🟥
Danish	554	🟥

[Choose Pre

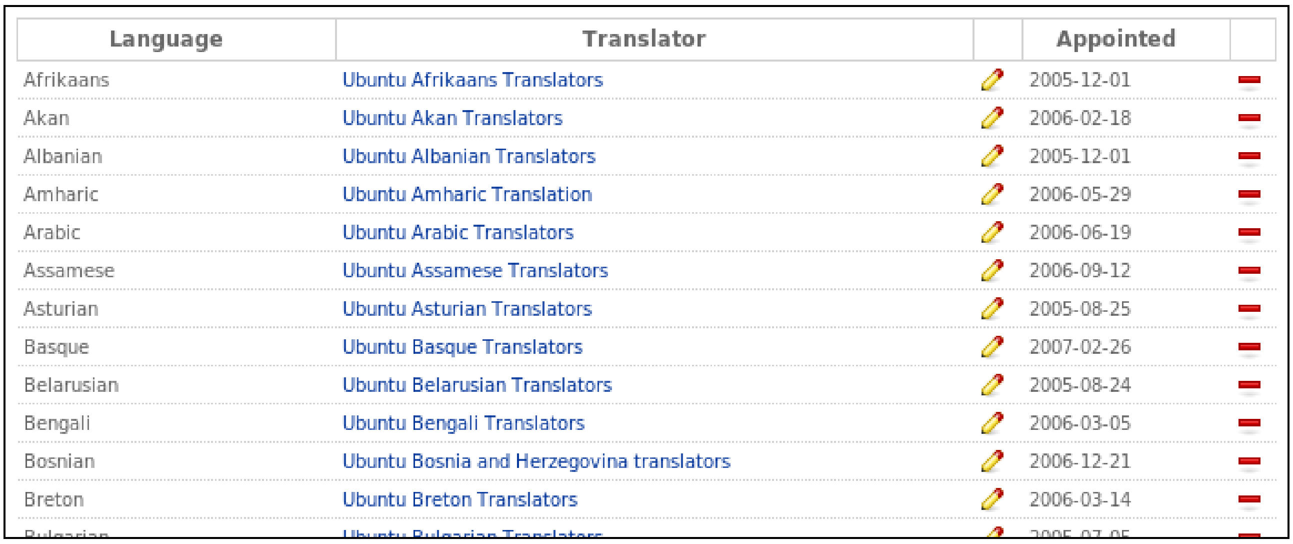
🟩 Translation unchanged since last synchronized
🟦 Changed in Rosetta
🟪 Newly translated in Rosetta
🟥 Untranslated

Translation teams

Perhaps most importantly, Launchpad makes it easy to organise your translation community. Here is a list of the translators and translation teams that work on Ubuntu. Of course, other people can contribute translations, but these are the formal teams who have the most direct access.

- https://launchpad.net/translations/groups/ubuntu-translators

At the time of translation, the list started with these teams:



Language	Translator		Appointed	
Afrikaans	Ubuntu Afrikaans Translators	✎	2005-12-01	▬
Akan	Ubuntu Akan Translators	✎	2006-02-18	▬
Albanian	Ubuntu Albanian Translators	✎	2005-12-01	▬
Amharic	Ubuntu Amharic Translation	✎	2006-05-29	▬
Arabic	Ubuntu Arabic Translators	✎	2006-06-19	▬
Assamese	Ubuntu Assamese Translators	✎	2006-09-12	▬
Asturian	Ubuntu Asturian Translators	✎	2005-08-25	▬
Basque	Ubuntu Basque Translators	✎	2007-02-26	▬
Belarusian	Ubuntu Belarusian Translators	✎	2005-08-24	▬
Bengali	Ubuntu Bengali Translators	✎	2006-03-05	▬
Bosnian	Ubuntu Bosnia and Herzegovina translators	✎	2006-12-21	▬
Breton	Ubuntu Breton Translators	✎	2006-03-14	▬
Bulgarian	Ubuntu Bulgarian Translators	✎	2005-07-05	▬

Note that each "translator" in this case is a team. They can also be individuals, but in our experience it is useful to create teams as soon as you have more than one person who is judged competent to help with a particular language.

One lesson that we learned the hard way is that it is better to set high standards for team membership than to invite too many people into the team before they have proven their experience and competence. Not only is it a question of speaking the language fluently, but it's also important to know the conventions of that application, and the details of such things as plural forms and data substitutions. To deal with these issues, we have implemented review systems that allow you to invite people to contribute translations, which are then held for review by the official translation team. This way you can mentor translators to improve, and ultimately accept them into the editorial community with permission to change any translation.

Levels of permission

You can determine the restrictiveness of your translation policy. There are three options:

- **Completely open:** anybody can fix any translation in any language.
- **Partly restricted:** for languages where a specific translator has been appointed, only they can made translations. For other languages, anyone can make a translation.
- **Restricted:** appointed translators can work on their specific language. For languages where a translator hasn't been appointed, anyone can suggest translations and they will be queued for review until a translator is appointed for that language.

In summary, Launchpad can be a useful tool to organise the people in your community who are willing to help make your software accessible to people in their own native language. You do still need to mentor people to ensure the quality of the translations, but Launchpad greatly reduces the barriers to participation and gives you a framework to build a strong translation effort.

The next step on our tour looks at Launchpad's feature planning and release management capabilities.

Blueprints: lightweight specifications

Planning isn't quite the least-favorite thing for most free software developers - but it's more or less in the same region as dealing with spam.

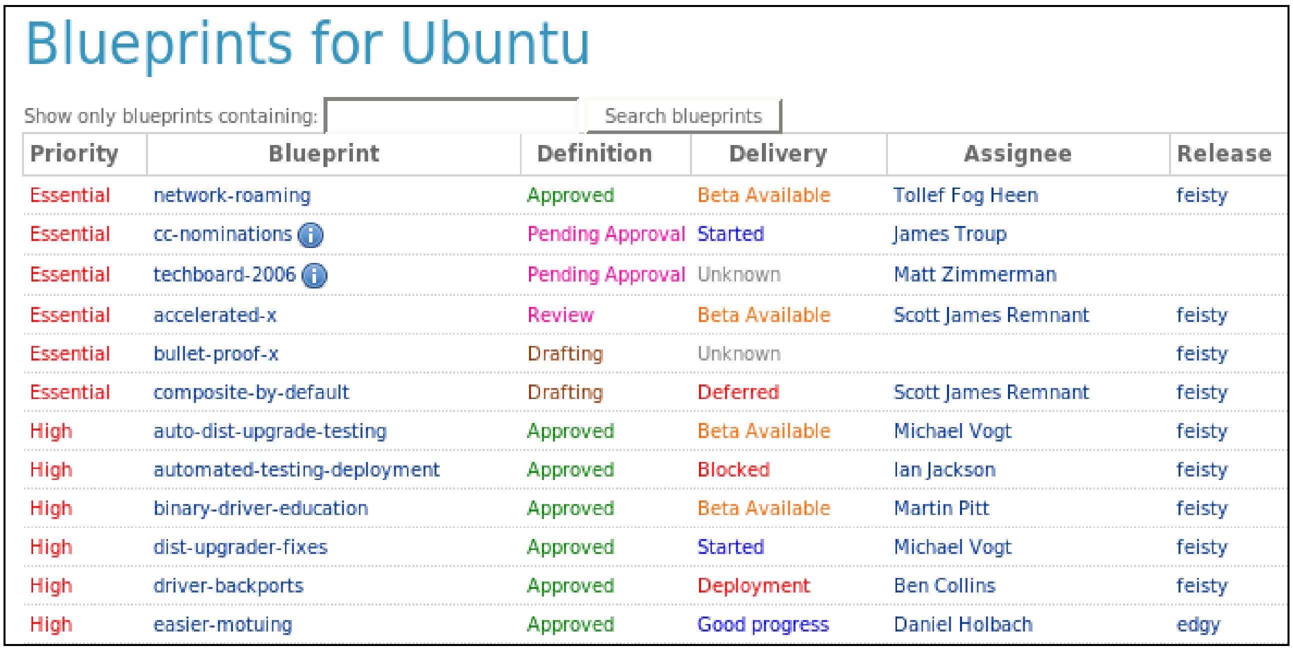
The free software process is highly agile, with release management policies that span the full range from "time based" (set the date and forget the features) to "when it's done" (set the features and forget the date). Launchpad doesn't aim to change the psychology of projects by imposing any planning, but it does projects to communicate internally, and with other projects.

Launchpad tracks placeholders for each chunk of work, idea for implementation or section of documentation that needs to be written. We call those placeholders Blueprints. A Blueprint can be as little as a single sentence, or a fully-fledged specification with data model and user interface mock-ups. The level of detail required is entirely a matter for the project to decide.

Each Blueprint belongs to a particular project. That means you can see lists of all the ideas, proposals or suggestions that are "out there" for a given project. For example, Ubuntu has more than 1,000 such blueprints, in various states of completion or discussion:

- https://blueprints.launchpad.net/ubuntu

At the time of writing, the most "important" of these blueprints were:



Blueprints for Ubuntu

Show only blueprints containing: Search blueprints

Priority	Blueprint	Definition	Delivery	Assignee	Release
Essential	network-roaming	Approved	Beta Available	Tollef Fog Heen	feisty
Essential	cc-nominations	Pending Approval	Started	James Troup	
Essential	techboard-2006	Pending Approval	Unknown	Matt Zimmerman	
Essential	accelerated-x	Review	Beta Available	Scott James Remnant	feisty
Essential	bullet-proof-x	Drafting	Unknown		feisty
Essential	composite-by-default	Drafting	Deferred	Scott James Remnant	feisty
High	auto-dist-upgrade-testing	Approved	Beta Available	Michael Vogt	feisty
High	automated-testing-deployment	Approved	Blocked	Ian Jackson	feisty
High	binary-driver-education	Approved	Beta Available	Martin Pitt	feisty
High	dist-upgrader-fixes	Approved	Started	Michael Vogt	feisty
High	driver-backports	Approved	Deployment	Ben Collins	feisty
High	easier-motuing	Approved	Good progress	Daniel Holbach	edgy

Blueprint status and priority

Each Blueprint has a priority, a "definition" status, and a "delivery" status.

The priority is set by project leaders. Anybody can contribute Blueprints for any project - there is no way to prevent someone from posting their ideas. However, the project leaders can set the priority - which means the extent to which they endorse the idea, or think it is important to implement soon.

The "definition" status tells you whether or not the project has reached consensus on how the idea should be implemented. In some projects there will be a person, or team of people, who will approve the plan. In others, plans are considered unnecessary or harmful, so this value is less important. In Ubuntu, we try to have a senior contributor review and approve any significant piece of work that is planned for any given release. Of course, lots happens without these plans, but it does give us some certainty that the various plans gel well, and that people have thought about the most important issues before they commit to getting something done in a particular release.

Finally, the "delivery" status is all about implementation and execution. It tells you whether the work has been done, or whether it is on track to be done.

Storing the detail of a Blueprint

Launchpad itself only contains a summary of the Blueprint - usually just the introductory paragraph - and then a URL to the location of the real document. In some cases, the single paragraph (or sentence) is enough, but it's more typical to keep the full document in a wiki, where members of the community can easily collaborate.

Just having a list of proposals and ideas in one place is useful, even if, as in the case of Ubuntu, there are clearly many more ideas than developers! It's convenient to be able to point new members of the community at a single place where those ideas are cataloged and to allow people to gravitate towards the pieces they are most interested in. People can subscribe to Blueprints and get notifications when their status changes and even when the wiki document they are in is updated.

Newcomers can easily see which ideas are important to the project leaders, and which are not, so they can choose to focus their contributions on those pieces most likely to be accepted into the project.

Linking Blueprints

Launchpad allows you to link a blueprint and a bug, or a blueprint and a branch. This allows people to see how pieces of work relate to one another.

It's very useful, for example, to be able to see the code that implements a blueprint evolving over time. It's also possible to link blueprints to one another, indicating rough dependencies. This lets you map out the order in which pieces should be implemented.

Release management

The most useful aspect of Launchpad's blueprint tracker, however, is the ability to group the blueprints that describe chunks of work that the project thinks are important to track for the next major series.

Here is the list for the Feisty (7.04) release of Ubuntu:

- https://blueprints.launchpad.net/ubuntu/feisty

As you can see, this is a much shorter list of around 100 blueprints, instead of 1100. These are blueprints that have been reviewed by the team. At he beginning of the Feisty development cycle, the team planned to implement those features.

In general, about 80% of the planned feature goals have landed in each Ubuntu release. We choose to ship on time, rather than necessarily waiting till every feature lands. However, different projects can adopt different release management goals.

The important thing, of course, is that everyone can see where we stand on any particular item.

Helping improve communication

The blueprint capability in Launchpad is not designed for hard-core project management. It doesn't offer Gantt charts or waterfall diagrams or block out developers hour by hour. But it certainly can improve the communication within a project, and between projects, about the status of work that has been discussed. Organising blueprints by release helps to communicate a clear picture of where any give release stands - what's in, what might get in, and what's not going to make it.

The release management that we've described here is at the level of major releases - what we call a series.

Sometimes you may want to group just the blueprints that are relevant for an interim release. For that, we'll use milestones. And learning about those is the final stop on our tour.

Milestones

We've discussed projects and series, which are the major ways in which we keep track of the progress of a free software project in Launchpad.

In general, most of Launchpad's tools allow you to group information at the level of the series. For example, different series in the same project can have different translations, and you can plan to fix bugs in a given series or implement blueprints in a series.

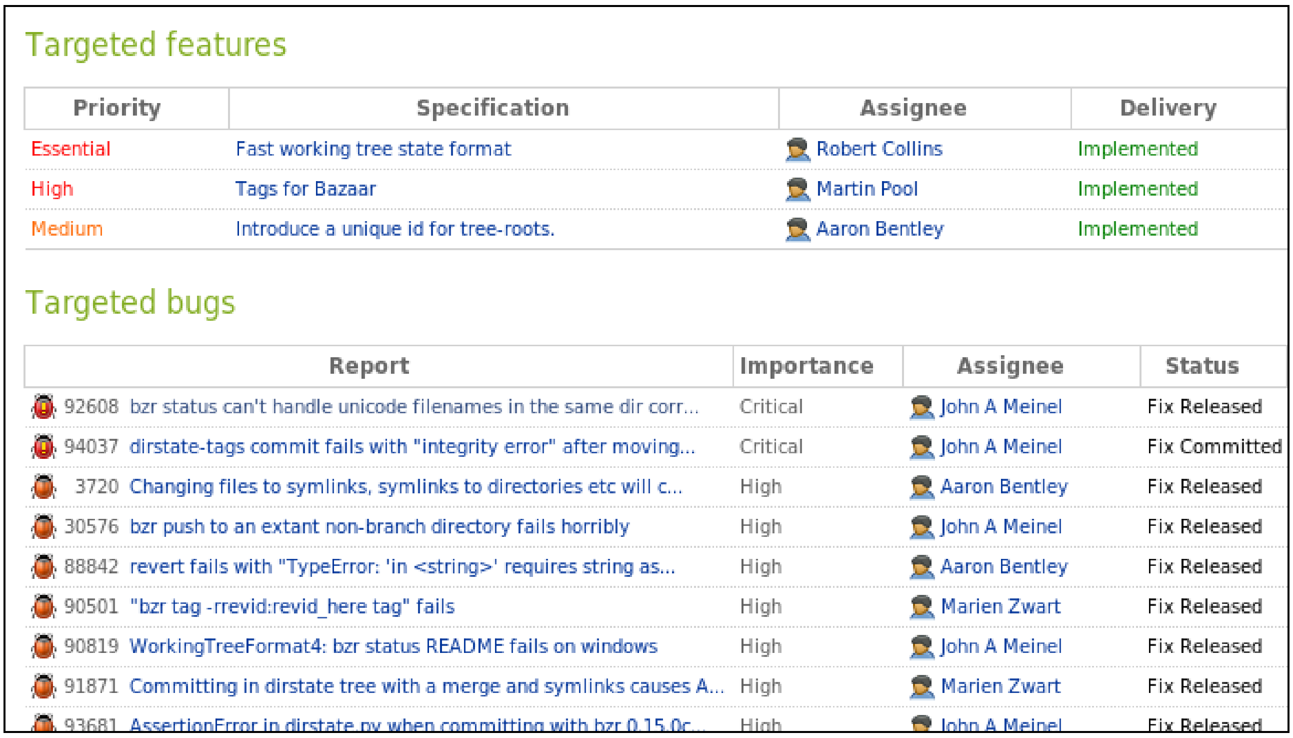
Launchpad allows you to have strict control over the bugs and blueprints that are planned for fixing or implementing in a given series. If you want to, you can require that a project driver approve each bug or blueprint that is linked to a particular series. In this way you can ensure that you are only making commitments that the project can keep. This is quite a heavyweight management approach, but it's useful for large projects like Ubuntu. Smaller projects often just turn that control off.

By contrast, milestones are a very lightweight way to organise a group of bugs or blueprints

A milestone is a point in time, or a test release, for which you need to keep track of a few bugs or blueprints. In Launchpad, you can easily create a milestone, and then link bugs or blueprints to that milestone as a way of saying "we think these items are worth keeping track of as we get closer to that date".

Here's an example, the 0.15 milestone of Bazaar:

- https://launchpad.net/bzr/+milestone/0.15



Targeted features

Priority	Specification	Assignee	Delivery
Essential	Fast working tree state format	Robert Collins	Implemented
High	Tags for Bazaar	Martin Pool	Implemented
Medium	Introduce a unique id for tree-roots.	Aaron Bentley	Implemented

Targeted bugs

Report		Importance	Assignee	Status
92608	bzr status can't handle unicode filenames in the same dir corr...	Critical	John A Meinel	Fix Released
94037	dirstate-tags commit fails with "integrity error" after moving...	Critical	John A Meinel	Fix Committed
3720	Changing files to symlinks, symlinks to directories etc will c...	High	Aaron Bentley	Fix Released
30576	bzr push to an extant non-branch directory fails horribly	High	John A Meinel	Fix Released
88842	revert fails with "TypeError: 'in <string>' requires string as...	High	Aaron Bentley	Fix Released
90501	"bzr tag -rrevid:revid_here tag" fails	High	Marien Zwart	Fix Released
90819	WorkingTreeFormat4: bzr status README fails on windows	High	John A Meinel	Fix Released
91871	Committing in dirstate tree with a merge and symlinks causes A...	High	Marien Zwart	Fix Released
93681	AssertionError in dirstate.py when committing with bzr 0.15.0c...	High	John A Meinel	Fix Released

Milestones are useful as a quick way to focus attention on a subset of bugs or blueprints which are important over the next week or month, before a specific point in time, or before a test release is made. They are not as rigorous in terms of control: anybody can throw a bug onto the list, or a blueprint. However, they are very useful for smaller projects, or projects that don't have the resources to plan major releases.

Thank you!

Thank you for taking the time to review all of these highlights in Launchpad! There is a lot more, beneath the hood, and it does take time to come to grips with an infrastructure this large.

Launchpad is constantly evolving, so please join us on `#launchpad` on `irc.freenode.net` or on the launchpad-users mailing list. We are always working to improve Launchpad for Ubuntu, and hope that it will be useful to other projects as well.