Fedora Package Maintenance - making Fedora and Red Hat developers more productive

(aka bridging the gap between our developer infrastructure)

Author: John Palmieri <johnp@redhat.com>

Impetus

Desktop Team Experience

According to various presentations given during Red Hat new hire orientation, Red Hat only hired A+ people. That is we hire people with such drive and creativity that they are all assets to the Red Hat team. However I have seen these smart people get bogged down with time sinks such as tracking upstream changes and constant packaging of new releases, bug fixes and security errata instead of what they do best - developing components for the next generation Desktop. Those who wanted to do more development flocked to projects like OLPC or Online Desktop.

Solutions include hiring people who have no interest in anything other than packaging and tracking upstream or producing tools which make this as easy as clicking a few buttons.

OLPC Fedora Experience

In OLPC we integrated packaging with the Fedora infrastructure soon after it became public. From those experiences we found much resistance from people because of the large learning curve in addition to the already high learning curve of creating RPM packages. Process, while needed, added time to all involved and did not help in getting people enthusiastic about being part of the Fedora ecosystem. Packages I did not actively herd through the process stayed in queues for months. For packages I tried to push through faster I inevitably missed or did not understand a process step during the long checklist of directives that needed to be fulfilled before a package could be added to the builds. If there were errors that needed to be corrected the overhead of the back and forth through bugzilla could mean a week or more before simple fixes were approved as correct. Add to that people becoming disenchanted with the process and just giving up and you realize how easy it is to lose or stall good contributions to Fedora.

Solutions include again hiring people whose sole job it is, is to marshal packages through the package system or consolidating the process in which it takes to get approved while providing integrated tools which provide explanation for each step in the process and the ability to semi automate the process. Having packages done right the first time reduces the amount of needed back and forth plus, if done right a process automating tool can be applied to other Fedora processes to make them smoother in the future.

Users

- Packagers - people who's goal is to create and maintain packages within Fedora
- Release engineers - people who create whole distribution using Fedora packages
- Package reviewers - people who review new packages destined for Fedora and ensure the quality of both the packages and the packager
- Enthusiasts - people who have not yet become part of the Fedora community but still want to experiment with newer packages and custom Fedora spins

Use Cases

Packagers

- Create a new package and push it through the package process
- Clean up and maintain packages
- Pull down and build newest upstream packages
- Pull down and build snapshot packages
- Manage patches (including pushing them upstream)
- Build scratch and mock builds
- Monitor builds in koji
- Organize packages
- Push packages through release processes
- Manage bugs filed on packages
- Handle errata process
- Adding and removing ACLs for owned packages
- Requesting ACLs for packages

Release Engineers

- Mirror repo's locally
- Integrated LiveMedia builds
- Tag, move, manage packages in repos
- Branch and repo creation

Package Reviewer

- Workflow checklist for package review with help and automation for each step
- Bugzilla integration
- Easy packaging for testing

Enthusiast

- Create local repos
- Automatic best guess packaging of unpackaged upstream tarballs
- Package install and revert capabilities for testing

Current Tools

Currently Fedora has a lot of tools for performing the above tasks. The biggest problem is that they are not integrated and you often have to go looking for them and log into different systems to use them. Bellow is a partial listing of those tools.

- Various editors - used to edit spec files, source code for diffs and other files. This is a free for all since everyone has there favorite editor, which means it is hard to simplify editing things like spec files. This causes formats to diverge wildly based on the person's own style. Editors also have little knowledge of the complete system. For instance updating a package requires knowledge of CVS, downloading upstream tarballs, uploading tarballs to Fedora repositories, editing the sources file, editing the spec file, kicking off a build in koji and pushing to Fedora. Adding these features to every editor our devs use would be futile.
- Pida (http://pida.co.uk) - This is a python IDE tool which is almost through the packaging process for Fedora and shows good promise as an integration point because of their plugin architecture, support for various text editors and integration with various bug systems and version control systems. More research is needed.
- Version control - used for collaborative editing of upstream code and Fedora's own packages. Various formats such as CVS, git and mercurial make this harder to support, not to mention the speedy pace of development on some of the newer formats.
- Koji - The Fedora project's build system tools. This is growing in leaps and bounds and becomes more and more usable as time goes on. It provides great command line tools along with a web interface and support for integration via web exported APIs and RSS streams. Problems include yet another place to login, not integrated with other web assets like the wiki and Bodhi. Various parts of the system are not integrated and require searching for the correct URL to do things like push packages once they are build. Little or no errata integration.
- Bodhi - The package database which handles pushing packages, new package submissions and handling security access to packages. Somewhat more immature than Koji but shows real progress. Biggest issues are no integration with Koji and completely independent development from koji. Again one must find the Bodhi url in order to use it and it is yet another place to login.
- Admin web - Place to admin users and groups and give people access to various Fedora resources such as version control. Again another login and not integrated with the rest of the web assets. Admin web seems to be a thin layer over the database and doesn't even support searching. It uses and is developed from a different framework than Koji or Bodhi so did not gain the benefits of search when the more user oriented and publicly visible resources gained them. Because of this and other issues, usability is dismal. This is understandable since it is not user facing but still needs fixing. Also it is confusing

when to use Bodhi (packages) and when to use admin web (users and groups) to assign access rights and a not so quick search of the wiki is often the result.

- Wiki - Where all of our online documentation of Fedora is added. This is hard to search, requires yet another login and is hard to edit (though much easier than straight HTML). Since this is the entry point for most new developers we need to streamline the wiki so that it is easier to find information and easier to contribute. Easy integration with our bug systems and other online assets would be nice also.
- Bugzilla and Trac - Our bug tracking systems. We also currently use these for package review. Yet another login with little integration with our other web assets and little integration with upstream bug tracking systems. For instance Bodhi package database should give me links to all bugs filed on a package I am looking at. The reverse should also be true. As for the review process, bugzilla is poorly suited because it requires manual upload of reviewed packages to outside servers (another login required by the packager) and long back and forths between the reviewer and the packager. While bugzilla could be used to track the packages progress, bodhi should be used for submissions and koji should automatically send srpms through our build systems to make sure they build on all architectures. I have seen many developers get fed up with all the manual process they have to go through just to get a simple package in.
- Package review process - Not quite a tool in the tradition sense it does however provide guidelines on the wiki for reviewing packages, making sure Fedora packages conform to a high standard. It is a long checklist that is prone to human error in interpreting each inspection point. It is also a daunting and exhausting workflow which causes reviewers to often take a long time getting back to packagers. Much of this can be semi-automated by scripting each process step and sending the package off to the next level once the review is done.
- Mock - Tool for creating clean buildroots to build packages in. This is used by koji to do builds but is hard to setup on local machines. It would be nice to be able to get an srpm and be able to build it in a mock instance with little setup (such as at the press of a button).
- rpmlint - A tool for checking if an rpm is packaged correctly. Right now it spits out warnings (sometime cryptic) but can be used to create tools that help correct bad spec files. It can also be used to make sure spec files are up to standards during updates and not just when creating a new package.
- rpmdev-newspec - Part of the rpmdev packages, this creates a new spec file from template when creating new packages. These tools are not integrated with anything and I had to be told about them before I used them. Newspec is dumb, copying over templates and filling in various values based on the type of spec file being created. It offers no help once the spec file is created. This is a start, but being able to edit the spec file with smart tool which understand the format, needs to be the goal.

Piecing Together A Solution (the overview)

It is clear the solution here is integration and improvement of our various assets. All of them have been evolving somewhat on their own and provide a great foundation on which to build on. I propose a three prong approach which in themselves can share code and be integrated with each other. These three are the web, client and scripting approaches.

Web

- We already have some great assets on the web. This is the future as it provides access from anywhere and from any device. We should continue to build out these resources, integrating them with one another and produce a tools home page which provides easy links and explanations for each of them.
- Single signon is very important to integration though I hear this is already in the works.
- There should be more code sharing or at least design sharing so things like search can be implemented easier across the board.
- Metadata and external APIs should be maintained and improved as integration points for the next two categories.

Client

- A well maintained client IDE is tantamount to tearing down the barrier to entry for new developers and streamlining access to the online tools (integration with XULRunner).
- By simply yum installing this one tool all of the developer tools needed to become a Fedora developer are also pulled in.
- The client IDE would be a centralized point for discoverability of all the other Fedora resources and also provide a deeper level of integration by providing links to resources based on what the user is currently editing (e.g. A link to the package database and past koji builds when editing a package spec file)
- The IDE would provide a central point of automation for package building (e.g. one button package updates from upstream, integrated spec file editing based on changes to upstream packages, etc.)
- Automation would extend to workflow (e.g. drive new package creation up to acceptance and future maintenance while stepping through the review checklist and updating the various bits in the review process)
- Koji integration would monitor builds and notify when they are done or failed.
- Offline tools for building when not connected to the Internet
- Management tools for mirroring local test repos
- The code would be modular so other tools can benefit from the deeper integration

Scripting

- Using the modules developed for the IDE one could create interesting python scripts to automate management. Examples:
    - Update all of GNOME locally and then push to koji once it has been tested
    - Compile reports on how often packages are updated
    - Monitor packages and run them through local tests when they are updated

    - etc.
- Scripts can also be added to other tools like the IDE
- Some people just like command line tools

Wiki

I thought I would throw this in as a future wishlist item to think about. Editing a wiki is hard and really should be more like word processing. AbiCollab, AbiWord's collaboration features shows much promise here. I have seen three or more people editing a document at the same time on the OLPC. I would propose repurposing AbiCollab so that people connect to the wiki server through AbiWord when editing pages. This is a lot of work as it means making AbiWord understand wiki formatting (or moving all the wiki pages to a new format) and providing fallback web editing for those who don't have access to AbiWord. It also increases the use of bandwidth per page being edited as well as taxing the servers which need to do much more processing.

Conclusion

This is just a high level proposal. Design and implementation is the next step. This document lays out the issues we faces and a possible solution that would make developing for Fedora a breeze. By bringing together all of the hard work we have already accomplished and lowering the barrier to entry we create a self feeding ecosystem which sees more developers jump on, which in turn creates a better Fedora and starts the cycle over again. It is essential to Red Hat's growth that we continue to grow the Fedora ecosystem and I believe these tools will do so by giving us an edge over our competitors.   The more our developers do not have to worry about the maintenance processes that currently bog them down, the more time they have to concentrate on innovation and improving the end results that users see.