How to improve the release process

This is the responsibility of the Packaging Team and the Release Team

This page is to discuss and plan on how to make Releasing easier and less time consuming. Most of them should not only be done at release time but also as part of the Continuous Integration. See also: Automated checks on the Tiki code base

Set better permissions in the the Tiki zip/tarball

- On some hosts, if you unzip the Tiki files and point your browser, you get an error 500 (ex.: files are write-able group) instead of the installer. So there is no error message or explanation of what to do next. If you run sh setup.sh, the permissions are set correctly, and you can proceed with the installer. How could we have the permissons set in the zip/tarball so at least the user gets some sort of usable feedback, and not an error 500. Site5 is an example (although it's out of date). Ask Marc for other examples if you can help resolve this. SourceForge.net permits to store "svn:executable" as seen here, but this may not be the place we want to manage this.

Check DB files

- ~~Script to automate what are now manual steps for Releasing (Drop Table, MyISAM, lacking _tiki.sql suffix, InnoDB), including in the schema patches~~ Scripts created and just need testing and to call from release script.

Check links

This is also something that should be added to Tiki as feature (on Tiki content). See Link checker

script to check external links

to doc.tiki.org

Done: Preferences report. Missing modules

to all other sites

Detect http:// and similar.

Script to check for internal broken links

- This would help us detect, and delete useless (and potentially unsafe) files.
- We already have a script which checks if php syntax is ok. What about broken links, like images?
  - also, as an example, tiki-register_site.php was removed as a link from the admin panel, presumably by accident.

Stats

- Would be good for marketing to report on how many commits by how many different people since the previous version. 3.2 vs 3.1 for minor. 3.0 vs 2.0 for major

Script to upload to SourceForge.net

- like LimeSurvey

Other ideas & script

- Test run with http://profiles.tiki.org/Test_All_Features to see if any features/plugins/modules are crashing Tiki
- php doc/devtools/release.php --testrun

- Improve handling of Security Check exceptions
  - security script: list of plugins with no validation?
- Reduce disk space
- A pref can't be present twice in the same tpl or it is impossible to change it
- http://closure-compiler.appspot.com will show warnings about JavaScript that may break (in old version of IE for instance), possibly could be automated via it's REST interface
- Tiki release script should indicate what libs were updated since last release

Package Integrity/Security

- Coordinate with Tiki Security Coordinator (Brendan).
- See https://sourceforge.net/projects/tikiwiki/files/Tiki_11.x_Vega/11.1/Verification/
- Pete hereby suggests, and offers to perform, these steps during a release:
  - svn co <branch>
  - Create a digitally signed listing of the secure hashes of _all_ of the individual files in the checked-out released branch.
  - Run the standard Tiki release script to create Tiki tarballs.
  - Create digital signatures and hashes of the four tarballs with the Tiki release key.
  - Upload tarballs, signatures, and lists of hashes to sf.net.

Digital Signatures

- Have the release packaging system automatically sign the 4 created archives (zip, gz, bz, 7z) _before_ uploading them to sf.net, preferably via a shell pipe (|) before even hitting disk.

Signed Digital Hashes

- Adding these authoritative fingerprints will help people to make sure the Tiki downloads have not been tampered with.
- The list of fingerprints should be signed with the Tiki release key.

Update "Current releases" Dynamic Variable

- Dynamic variable "Current releases:" %stablereleasenumber% in info.t.o (and elsewhere?)

Discussions

To branch or not to branch

- When to branch

To merge or not to merge

Benefits of backporting from trunk to branch

For example branch 4.x would be something like "pre-proposed" where quality team can decide to rollback if anything inappropriate gets in, but they don't need to approve each and backport themselves to stable 4.x yet. At time of 4.0 release, this branch will become stabilized (what 3.0 is now) and new proposed branch for 4.x will be created...

- avoids merge pain to trunk
- devs can decide what code they think works well tested in trunk and would like to see released and can backport themselves
- only tested things (bugfixes, UI enhancements) get in from trunk (no new features after feature

freeze/branch creation) and we can be more sure we don't release weak / untested / unstable
- possibility to clean and KIL as needed in the branch 4.x before final release without loosing code people work on in trunk but don't make it in time (keeping things on track in trunk)
- we don't switch the merge/backport stream all the time before main release comes and don't force people to swicth to work on another branch

Accept	Undecided	Reject
2	0	0
• luci • sylvieg_if_we_have_the_ressource		

Benefits of merging from branch to trunk

- faster development and focus on release (when devs are forced to switch to 4.x branch)

Accept	Undecided	Reject
1	0	1
• sylvieg_if_we_do_not_have_the_ressource		• luci

Related links and tools

- If you find yourself writing custom scripts to handle the packaging, deploying, or testing of your applications, then we suggest looking at the Phing. Phing comes packaged with numerous out-of-the-box operation modules (tasks), and an easy-to-use OO model to extend or add your own custom tasks.
  - While this seemingly rudimentary implementation may not appear as “powerful” as Phing definitions, its simplicity belies its incredible flexibility. It leverages your existing knowledge investment in PHP, and with a little creativity and imagination you can use Composer’s dependency resolver and native PHP scripting to create some fairly complex build and take-down tasks. You could even integrate this into Jenkins for continuous integration.
- https://github.com/liip/RMT/blob/master/README.md. Very cool!

Phar archives

- http://www.slideshare.net/helgith/phar-the-php-exe-format
- http://stackoverflow.com/questions/3521484/advantages-of-php-5-3s-phar-archives
- http://www.ibm.com/developerworks/opensource/library/os-php-5.3new4/index.html
- http://phpmaster.com/packaging-your-apps-with-phar/

Notes

- We would need a clean / clean / separate place to put db/local.php and custom theme
- What about cache?
- Easier download on FTP
- Can make signature of the file, so sure it's not changed

Big picture

Each release is three big things:

1. **Packaging (running the scripts, distributing and promoting the release)**

    - As described at Releasing (except for the "Fix major bugs and regressions" section)
    - This is already quite well streamlined and there are some ideas higher on this page to make this even better
    - Minor releases (ex.: 9.3) take very little time because it's mostly just this, and not the next two parts
2. **Fixing bugs, cleaning up, doing all the stuff that was kept for "later"**
    - Ex.: A dev starts something, disappears, and someone else has to clean up the mess
    - To get an idea of the workload, consider that not all regressions get fixed: Regressions in trunk, Regressions in 8x, Regressions in 9x, Regressions in 10x, Tiki9, Tiki10
    - This, in fact, is the part which takes the most time. If the release was every 12 months instead of 6 months, we would, in fact, have more than twice the workload here because the longer we wait, the harder it is to fix (developer no longer available, other code is intermingled so clean rollback is impossible, etc.)
    - Minor releases: They contain very few regression risks and thus, there is very little to do here
3. **Updating all *.tiki.org to the soon-to-be-released version**
    - We need more Teams to be active and to care for their site outside the releases and they should naturally tend to them during the upgrade process
    - This takes a lot of time because we deal with issues that are in #2 above but weren't caught yet (or they were seen but since they weren't affecting a *.tiki.org site, they didn't get fixed)
    - Minor releases: All *.tiki.org sites are updated daily (except tiki.org which is on demand) to the latest code for the branch and thus, there is nothing to do for minor releases.

See also Release Coordinator , Release Roles and Pre-Dogfood+Server.

To catch / fix issues and regressions sooner and make *.tiki.org upgrades easier, dev.tiki.org (and maybe themes.tiki.org or doc.tiki.org) could run on monthly updates of trunk code, while nextdev.tiki.org is daily trunk. Before each monthly upgrade, nextdev.tiki.org is checked to catch and fix regressions. This would make sure to keep trunk always closer to "release at any time" state. These releases could be called Tiki11DR1, Tiki11DR2 or Tiki11pre-alpha1, Tiki11pre-alpha2, etc.

For some of the *.tiki.org sites to run on a monthly snapshot of trunk

Accept	Undecided	Reject
0	0	0