Version lifecycle

This page needs an update because of the changes decided at Quality Team - How to reduce the workload, mainly that we no longer use proposals branches.

Goals

Merged into: http://info.tiki.org/Version+Lifecycle#Goals

Background information

Since Tiki2, we follow time boxing approach like Gnome and Ubuntu, with a new major release every 6 months (October and April). Whatever isn't ready in time is deferred to the following release. There are several branches (Old stable, Stable, Dev, Experimental, Legacy)

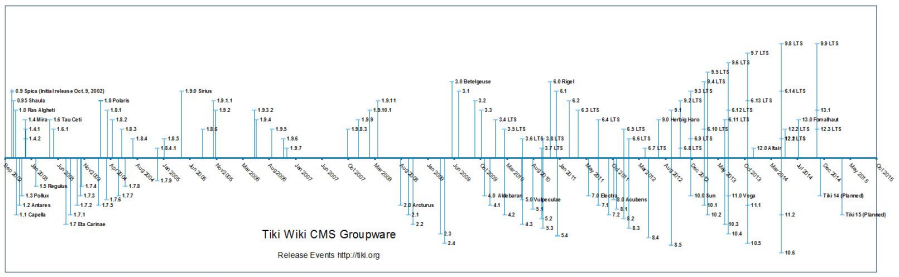
If a change requires more time to implement, we break it into multiple parts and work on each chunk in each cycle. Workspaces is an example of a major change taking place over 2 versions.

Since all features are bundled in Tiki (vs having hundreds/thousands of extensions/plugins), we have inherent synchronized releases of all features.

When do we stop making bug and/or security fixes on previous versions? The goal is to make upgrades easy, but in many cases, site admins (ex.: in the Enterprise) do not want to upgrade... but they still want to get security fixes. If we tried to support for 2 years, we would have 4 versions to support plus the upcoming one! This would mean that a bug fix would have to done on up to 5 versions. And since code changes, the bug fix can't just be blindly copied. It must be tested. Reality check: As a community, we have to be very realistic about how much energy we'll have for this. Developers tend to be interested in working on new versions.

Thus, it was decided to mark certain versions as Long Term Support, and converge our energies on these. Tiki 3.x was the first official **long term support** version. 6, 9 and 12 are also LTS.

Background reading: Meta-cycles: 2-3 year major cycles for free software? and the follow-up: 2 year cadence for major releases: some progress



Tiki releases, to date

Release & branching strategy

7.x 2011-04

Tiki7 follows the same general process as Tiki 6 except that

- everything is 6 months later
- branches/6.x is **not closed**, as it will become LTS

Release 7.1 (May 2011)

- ~~Tiki 6.x (which is 7-months old by now) becomes LTS~~
- ~~Release one final 3.x LTS~~
    - ~~Tiki 3.x is closed. (around May 2011, thus 2 years after 3.0 release)~~
    - LTS users will upgrade (for example) from 3.7 to 6.3
        - For data, it will be easy thanks to the Database Schema Upgrade and already widely tested because everyone in the community has gone though that process over last 2 years.
            - This being said, testing the direct 3.x to 6.x/7.x upgrade is important because there could be some issues only visible when you do such a jump.
        - For customizations however, they most likely need to be redone either in similar hacking way or by using the newly available possibilities.
- Semi-automatic merging period ends and Quality Team goes in "strict mode". If you want to fix something for an eventual 6.4 LTS
    1. commit to trunk (for 8.0)
    2. commit to proposals/7.x (for 7.2, etc.)
    3. commit to proposals/6.x LTS (for 6.5, etc.)

8.x 2011-10

Tiki8 regular dev period (April -> September 2011)

- Everyone commits to trunk (while avoiding major changes in trunk that either won't be ready for the next release or are not intended for the next release). Use Experimental branches instead.
- If you have changes to trunk (typically changes to a large number of files) that will increase the difficulty of the 7.x -> 8.x Semi-automatic merging period, please do them after the merges are stopped (usually about when 7.1 is released)

As 8.0 is approaching (September 2011)

1. Pre
    - Solve outstanding issues
        - Merge all experimental branches
        - Remove any code that should be -> Endangered features
    - Do a run of the "Preparatory work" of Releasing (better to do before branching)
2. branches/8.x is started
3. Some *.tiki.org sites are migrated to branches/8.x
4. Institute semi-automatic merging period from branches/8.x to trunk] (future 9.x)
    - Everyone commits to branches/8.x (unless it's only for 9.x)
        - Merge from branches/8.x to trunk (future 9.0) is handled by script so you can commit to

branches/8.x until 8.1 is released. All devs should try to update the sites they manage during this period, because the process is simpler than after 8.1

8.0 is released (October 2011)

1. All *.tiki.org sites are running branches/8.x
2. Quality Team starts to check all commits and rollback any issues. "soft mode" -> dogfooding Code Review
3. Release one last 7.x stable (ex.: 7.4) with everything that is in proposals/7.x (which is closed after)
4. Merge from branches/8.x to trunk (future 9.0) is handled by script so you can commit (with extreme caution!) to branches/8.x until 8.1 is released. All devs should try to update the sites they manage during this period, because the process is simpler than after 8.1

Release 8.1 (November 2011)

1. branches/7.x is closed, as all focus goes on branches/8.x
2. proposals/8.x is created and Semi-automatic merging period ceases.
3. Quality Team goes in "strict mode".

If you want to fix something for an eventual 8.2:

1. commit to trunk (for 9.0)
2. commit to proposals/8.x (for 8.2, etc.)

If you want to fix something for an eventual 6.5: (a lot of work!)

1. commit to trunk (for 9.0)
2. commit to proposals/8.x (for 8.4, etc.)
3. commit to proposals/6.x (for 6.5, etc.)

9.x 2012-04

Tiki9 follows the same general process as Tiki 8 except that

- everything is 6 months later

Since Tiki 9.x will be the next LTS, if you have a major change to make, avoid making it for Tiki9. After 9.1 is the best time to introduce major architectural changes for Tiki10, as it won't interfere with merge up script, and there is still plenty of time to deploy everywhere.

10.x 2012-10

Tiki10 follows the same general process as Tiki 9 except that

- everything is --6-- 8 months later
- branches/9.x is **not closed**, as it will become LTS

Release 10.1 (~~November 2012~~ January 2013)

- Tiki 9.x (which is 9-months old by now) becomes LTS
- Release one final 6.x LTS
    - Tiki 6.x is closed. (around November 2012, thus about 18 months years after 6.0 release)
    - LTS users will upgrade (for example) from 6.7 to 9.3
        - For data, it will be easy thanks to the Database Schema Upgrade and already widely tested because everyone in the community has gone though that process over last 18 months.
        - For customizations however, they most likely need to be redone either in similar hacking way or by using the newly available possibilities.
- Quality Team goes in "strict mode". If you want to fix something for an eventual 9.4 LTS
    1. commit to trunk (for 11.0)
    2. commit to proposals/10.x (for 10.2, etc.)
    3. commit to proposals/9.x LTS (for 9.4, etc.)

11.x and beyond 2013-04

Since Tiki2, the release process has improved dramatically and the Continuous Testing Server is expected to further improve the situation.

Upgrade paths examples

Moved to http://info.tiki.org/Version+Lifecycle#Upgrade_paths_examples

Benefits

- Only 3 branches are maintained by developers at any given time, and only one that accepts significant changes.
    - Permits to focus eyeballs: Anything that is not risk-free (ex.: new features, or re-factor) is always done in trunk, thus all eyeballs are focused in the same place. So, if a change has a side-effect, it can be caught early. We don't fall in the dependency-hell trap.
- Provides homes both for:
    - Fast pace of development
    - Stability and limited upgrades
- We can still make quick .x releases on stable and LTS branches for security fixes, bug fixes, and changes to APIs. Ex.: A reported change in the Facebook API

Drawbacks

- Previous branch (if not LTS) is dropped as soon as a .0 is released. Ex.: Once 5.x is released, 4.x is no longer maintained.
    - Thus, if you use 3.x LTS, and you decide to upgrade to 4.x, you may have regressions. If you want to upgrade from LTS, it should be to latest stable at the current time of the upgrade. This one will contain all the fixes that LTS has.
- Except for LTS, versions are only supported for 6 months.
    - If you need Long Term Support on other versions, you can always get professional support from Consultants.
- While it's OK to add completely new features, it is best to avoid major infrastructure developments (the ones that have a chance to introduce a lot of bugs/changes throughout the application) for LTS versions (like 6 and 9)
- While in the semi-automatic merging period, sweeping changes (inc. cleanups) on trunk must be avoided (about 4-8 weeks per 6-month period)

Longer LTS period

For many years now (and for the forseeable future!), Tiki has been the FOSS Web Application with the most built-in features. For many projects, upgrading to a new version is an annoyance because the existing version does everything needed and more. Because the 6-month release cycle was too frequent for some in our community, LTS versions were introduced. But even 18 months for LTS is also too frequent for some, so it is proposed to have a longer LTS period for security fixes.

Benefits

- You can start a project at any time and be assured of at least 2 years of security releases. (If you are lucky in your timing, you may have up to 3.5 years of support from your initial installation). Once you get in the LTS cycle, you can upgrade every LTS (18 months) or skip an LTS and upgrade every 36 months (which is also good timing for a new design and to change the OS)
    - At any time, you can upgrade to the regular 6-month cycle (if you need new features)
- Better for inclusion in Debian and Ubuntu LTS repositories
- Great for the enterprise
- Great for major projects which want to use Tiki more as a Framework.
- Great for web design shops which make highly customized designs. By the end of the support period, it will be time to make a new site design anyway.
- Starts LTS phase later so Quality Team has less work

Drawbacks

- More work
- Risk of dispersing community energy on more versions

Diagram

Moved to: http://info.tiki.org/Version+Lifecycle#Version_Lifecycle_Diagram

Strategies

Moved to: http://info.tiki.org/Version+Lifecycle#Strategies

Things to discuss / be careful about

- What is policy of included code?
  - Zend Framework
  - Smarty
  - jQuery
  - and other External Libraries
- What if a future version of a Tiki requirement (PHP/MYSQL/Apache) breaks something during the LTS period?
  - PHP 5.2.x is no longer supported but still used by a lot of hosts. Tiki works well on PHP 5.2 and 5.3
    - We should test with PHP53Compat_CodeSniffer
    - Tiki9LTS is supported for the PHP/MYSQL/Apache version which were available when it was being developed. ex.: Tiki9 LTS is widely tested in 5.2 and 5.3 It is expected to work in future versions, but if PHP (ex.: 5.5) makes some non-backward compatible changes, the LTS version will not be adapted to work with it.
  - Tiki 9 installs fine on MySQL 4.1.22 (although some of the upgrade scripts require MySQL 5) so it's safe to assume that future versions of MySQL will be fine
  - We should perhaps suggest to pair this with LTS servers and browsers?
- What if a future version of a major browser breaks something during the LTS period?
  - This is very unlikely but if it happens, we'll evaluate on a case by case basis.
- What if a bundled external library has a security issue and it's no longer supported?
  - We'll inform users about the risks and that they should discontinue the use of the features that require this library.
- Update notifier
  - What about Auto-installers?
  - This should help getting us in various distros?
- How do we deal with documentation?
  - Maybe the Preferences report script could indicate which versions have the given feature?
  - Perhaps we should try to have more docs in the app (easier said than done)
  - Use PluginVersions
- Make sure that demo.tiki.org has all the supported LTS versions
- Experimental features: preferences tagged as experimental in tiki-admin.php should not be covered by LTS
- How do we indicate that profiles are not supported for 5 years?

Definition of "security-only" phase

moved to: http://dev.tiki.org/Where+to+commit#Definition_of_security-only_phase

LTS: Proposal to extend security-only support period

We already offer 3.5 years of support. This permits to skip an LTS, so you can go from 6LTS to 12LTS. Going forward, let's make it possible to skip 2 LTS version. So Tiki 12LTS could skip 15LTS and 18LTS, and upgrade to 21LTS. Thanks to our model, the workload for this is quite reasonable. In other web apps with all the extensions, it would be a prohibitive work load.

The Tiki model is arguably already the most effective solution at solving the synchronized releases challenge in the web app world (all other systems have tons of extensions which are chronically late to update). With this proposal, Tiki will not only be the FOSS Web app with the most built-in features and the fastest release pace of the last 5 years, it will also be the one with the longest support period going forward.

- Ubuntu offers 5 years
- CentOS/RHEL offer 10 years
- Drupal supports current and previous version and releases are not scheduled, but historically approximately every 3 years. So if you started at the very beginning of a cycle, you could theoretically get 6 years of support, but in practice, real world sites wait 1-2 year for enough modules to be ported to the new version. So the real support period is more like 4-5 years. Since Drupal 6 support ends when Drupal 8 is released, and when Drupal 8 is released, not enough modules are ported, it is not possible to skip versions.
- Joomla! LTS: 2 years and 3 months
- MediaWiki LTS: 3 years
- WordPress project leader is "philosophically opposed" to LTS
- Typo3 LTS: 3 years
- Symfony: "three year period for bug fixes, and for a four year period for security issue fixes."

Accept	Undecided	Reject
1	2	6
• dirschneid	• marclaporte • xavi	• daniam • lphuberdeau • arild • amette • Torsten • pascalstjean

Adding 1.5 year security-only support period to LTS version starting in Tiki6 LTS (so a total of 5 year support)

Accept	Undecided	Reject
0	5	1

	• daniam • marclaporte • lphuberdeau • arild • xavi	• pascalstjean

Adding 1.5 year security-only support period to LTS version starting in Tiki9 LTS (so a total of 5 year support)

Accept	Undecided	Reject
6	1	0
• marclaporte • lphuberdeau • arild • xavi • pascalstjean • dirschneid	• daniam	

Adding 1.5 year security-only support period to LTS version starting in Tiki12 LTS (so a total of 5 year support)

Version Lifecycle Diagram

This info was moved to info

Related links

- http://linuxlifecycle.com/
- http://en.wikipedia.org/wiki/Software_configuration_management
- http://www.mediawiki.org/wiki/Version_lifecycle
- http://drupal.org/handbook/version-info
- http://fedoraproject.org/wiki/LifeCycle
- http://docs.joomla.org/Release_and_support_cycle STS: 7 months LTS: 21 months
- http://wiki.ubuntu.com/TimeBasedReleases
- http://blog.chromium.org/2010/07/release-early-release-often.html
- https://wiki.mozilla.org/RapidRelease
- http://www.leanessays.com/2011/07/how-cadence-determines-process.html
- Software G Forces: The Effects of Acceleration (video)
- O'Reilly MySQL CE 2010: Brian Aker, "State of Drizzle" -> very fast release
- http://blog.mozilla.com/channels/2011/05/13/firefox-5-beta-availability-and-add-on-compatibility/
- Where to commit
- Releasing
- When to release - think popcorn
- RoadMap
- Freeze and Slush
- Model
- http://symfony.com/doc/master/contributing/community/releases.html (very similar to Tiki!)

- Do Faster Releases Improve Software Quality? An Empirical Case Study of Mozilla Firefox.
- http://www.h-online.com/open/features/Comment-1-for-rapid-releases-1916446.html

Alias

- LTS
- Long Term Support
- Meta-cycles
- Release schedule
- cycle
- lifecycle
- release cycle
- extended support
- extended LTS
- Maintenance Policy
- Version lifecycle to