When to make major changes

As of 2021-04-29, this is a draft. Over the next few weeks, please participate to this thought process and once the community has spoken, this will become our new official guidelines on the matter.

Where to commit explains well when and where to make bug fixes and add features. But when and where is an appropriate time to make major changes? (refactoring, code cleanup, increase requirements, change Composer dependencies, etc.). *Where* is easy: it's always in trunk (perhaps after an experimental branch). But when? It depends.

In light of:

Info in related pages

- Versions
- Where to commit
- Semi-automatic merging period
- When to branch
- When to release - think popcorn
- How to release
- Freeze and Slush
- Git cherry-pick

Definitions

- LTS: A Long Term Support version as per Versions. Ex.: Tiki21
- LTS+1: the first major stable version after an LTS. Ex.: Tiki22
- LTS+2: the second major stable version after an LTS. Ex.: Tiki23
- There is no LTS+3 because it's time for another LTS.
- In the context of this document, we are mostly talking about the development cycle of the version. So LTS+2 starts the day the 23.x branch is created, even before 23.0 is released.

A version has phases:

- branch (from trunk)
- x.0, x.1, x.2, etc.
- EoL

Community dynamics on backports and LTS versions:

- Many community members prefer to run on LTS (for the stability), but they still want bug fixes and a some innovation. And they contribute as per Where to commit. So backports are done.
- Backports are the most risky/time-consuming
  - May need to backport to more than one version. For each commit, the cycle is repeated. It's easy to get mixed up. You may need to commit to LTS+2, then backport to LTS+1 (which you don't use), to then be able to backport to LTS (your goal)
- There is a period that the branch and trunk have very similar code bases, so backports are easy. To ease backports, for a while we have a Semi-automatic merging period. But after a while, it's too much work/risk, and we stop, and backports are done manually.
- Some LTS+1 are massive disruption
  - Tiki7 Trackers Revamp
  - Tiki13: Bootstrap 3 integration
  - Tiki19: Bootstrap 3 to 4 transition
- But others are pretty incremental, and are pretty much like a LTS+2

    - Tiki10
    - Tiki16
    - Tiki22

There is a symmetry above but this is not planned. Each LTS+1 was an opportunity for massive changes and it just so happens that innovations at that time where not disruptive.

Here are some guidelines:

Type of major change

Mass cosmetic cleanups of the code

These have a low risk to introduce issues, but they make backports difficult/risky because the code structure changed.

Examples:

- Moving to https://www.php-fig.org/psr/psr-12/

Good options

- Outside of the Semi-automatic merging period
- 4-6 weeks before branching, so it gives time to fix minor issues in trunk before branching
    - Any version works but likely a tiny bit better to do for LTS +2 because there are fewer backports than LTS+1.

Increasing requirements

This should be announced ASAP (so the community gets ready) and done early in LTS+1 development cycle do developers take advantage of new capabilities. Can also be done for LTS+2 but it should be announced before LTS+1 is released so users without those requirements stay on LTS. Best to avoid doing in LTS so everything has ample time to stabilize.

What we don't want: someone upgrades to LTS+1 and PHP version is OK. But then PHP requirement changes and they can't upgrade to it, and LTS+1 goes out of support.

Removing the use of deprecated features

As PHP versions evolve, we get advanced warning that some features will eventually be removed. They continue working for a while with warnings.

This can be done at any time outside of the Semi-automatic merging period. The sooner the better.

Often, out challenge here is not so much Tiki code (we just fix it), but the Composer dependencies which may be lagging. So this forces us to change libs, which increases workload and risk. But new lib will also bring opportunities.

Major system-wide changes

aka Breaking Changes. Ex.: Move to Bootstrap 3 or 4 or 5.

- This should be done early in LTS+1 development cycle.
- This will be a ton of work
- This will be a ton of changes
- There will be a ton of issues

These versions are known to be of lesser quality and tend to be avoided for larger projects (where it's difficult to test everything). But such users can just stay on LTS.

This is necessary to innovate.

So with this logic, the move to Bootstrap 5 will happen in Tiki25