Tiki Lifecycle

Get familiar with version policy and release roadmap

Tiki has a time-based, "Release early, Release often" approach with Long Term Support (LTS) versions, to balance the needs of stability and innovation.

Summary

- There is a new major version (e.g., 18.x 19.x) every 8 months.
- Minor versions (18.1 18.2) are released as needed, typically every one to two months.
- You can upgrade from any supported version to any subsequently supported version at any time.

However, there is no downgrade path. Please make sure you make a backup (of the database and any customized files) first.

- Some can have concerns about upgrade difficulty but this isn't the case because if you skip ahead several versions, you are going through the same path as everyone else did in the previous years, but in an accelerated fashion. You can upgrade from Tiki 1.9.0 (released on 2005-04-27) to any later version of Tiki because all the upgrade scripts are part of the code base. This policy will be maintained to keep upgrades as easy as possible.
- The support period for non-LTS versions is at least until the x.1 release of the next major version. For example, Tiki 19.x was supported at least until Tiki 20.1 was released.
- Every third version is a Long Term Support (LTS) version.

- The LTS support period is five years (since Tiki 12.x LTS), and consists of a period of ongoing features support (bug fixes and minor, self-contained enhancements) followed by a "security fix only" period over this time frame.
- The LTS version is expected to have minimal changes, no regression bugs, and little or no change in Tiki behavior, along with continued security fixes for an extended period of time. It's a good idea to install LTS versions on a similarly long-term supported server environment.
- Once a Long Term Support (LTS) branch becomes stable (x.0 has been released), the previous LTS (which is 2 years old by then) goes into "security fix only" mode. Users of older LTS versions typically prefer fewer changes (which they may need to audit) over bug fixes (which can introduce other bugs). Trusted contributors are allowed to contribute non-risky bug fixes (which typically emerge from enterprise projects).
- You should not expect LTS versions to be updated to solve an issue introduced by a newer PHP version that is subsequently released. However, issues of this type have historically been very rare and a patch to resolve such an issue would be analyzed by the development team.
- LTS versions are typically used for Enterprise deployments or by other users that do not want frequent major upgrades (for example, if your environment requires software audits before every software installation).

- Since Tiki has an all-in-one model, all features are released at the same time, and you don't need to wait for third-party extensions / plugins / modules to be updated (which is very common and often painful in other web applications). See also pluginproblems.com.
- Extended Security Maintenance is available from EvoluData

Goals

- Have a predictable system.
  - At download time, users should know until when the version is supported.
  - Time-based releases permit community members to self-organize to make sure specific developments are ready in time for the release.

- Have a rapid release cycle.
  - To take advantage of new technological innovations
  - Encourages contributions to the main code base as contributors see their enhancements rapidly become available as part of the next release, where they are tested, documented and improved.
- Cater to diverse needs, so the community converges efforts.
  - Users that want to have early and frequent access to the latest and greatest, and experimental features.
Users that want a supported version (with security fixes), with minimal upgrade effort / change, typically, enterprise users.

- Make it Easy to manage.
  - Make it easy to explain and understand.
  - Manage as few branches as possible. You will see in the chart below how branches overlap.

Why predictable and rapid are both important release characteristics

Feature enhancements are often the fruit of sponsored feature requests, or for the need to deliver a specific project. Site managers or IT departments can do enhancements themselves or choose to hire Tiki Consultants. The vast majority of the time, the sponsor is happy or insistent that their enhancements become part of the official code base. However, the timeline has to be reasonable. By having a set date (which is not too far off), there is an incentive to commit to trunk (the development version which will eventually become the current stable version), and know when it will become stable. Otherwise, customer/project pressure will make them fork their version and the community will often lose those contributions. They might plan to upstream later, but the later never happens, or in the meantime, someone else implemented the feature in a different way, and it becomes very difficult to converge.

Here is a recap of innovations in past years:

- 2009-05: Tiki3 LTS
- 2009-11: Tiki4
- 2010-06: Tiki5
- 2010-11: Tiki6 LTS
- 2011-06: Tiki7
- 2011-11: Tiki8
- 2012-06: Tiki9 LTS
- 2012-12: Tiki10
- 2013-07: Tiki11
- 2013-11: Tiki12 LTS
- 2014-08: Tiki13
- 2015-05: Tiki14
- 2016-04: Tiki15 LTS
- 2016-11: Tiki16
- 2017-07: Tiki17
- 2018-01: Tiki18 LTS
- 2018-11: Tiki19
- 2019-06: Tiki20
- 2020-03: Tiki21 LTS
- 2020-11: Tiki22
- 2021-08: Tiki23
- 2022-03: Tiki24 LTS
- 2022-12: Tiki25
- 2023-07: Tiki26
- 2024-04: Tiki27 LTS

The plan ahead:

- 2024-10: Tiki28
- 2025-06: Tiki29
- 2026-02: Tiki30 LTS

Not only has Tiki been for many years the FLOSS Web Application with the most built-in features, Tiki is
also the one of the FLOSS Web Applications with the fastest release cycles.

A huge part of our user base does not want or need new features, but only bug fixes and security fixes. This is why Long Term Support (LTS) versions were introduced in 2009. Even if they don't need the new features now, site admins are pleased to know that they will have easy access to them when they upgrade.

The cycle

- A new major version every 8 months (February, October, June, etc.).
- Minor versions are released as needed, usually every 6 to 8 weeks.
- Every third major version is a Long Term Support (LTS) version (Tiki 9.x, 12.x, 15.x, 18.x, 21.x, ...).
  - So the LTS cycle is two years (3 x 8 months = 24 months).
- Support period
  - LTS versions are supported for five years (starting from Tiki12).
  - Non-LTS versions are supported until the x.1 release of the next major version (exceptions might be made to delay this to the x.2 version if the x.1 version is released very close to x.0).

Version Lifecycle

If you need a longer support period, Extended Security Maintenance is available from EvoluData.

<table>
<thead>
<tr>
<th>Version</th>
<th>Release of x.0</th>
<th>Regular Feature Support (Security Fixes only thereafter)</th>
<th>End of Life</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.x LTS</td>
<td>2018-01</td>
<td>at least until 2020-01</td>
<td>2023-01</td>
</tr>
<tr>
<td>19.x</td>
<td>2018-11</td>
<td>until End of Life</td>
<td>not before the release of Tiki 20.1</td>
</tr>
<tr>
<td>20.x</td>
<td>2019-06</td>
<td>until End of Life</td>
<td>not before the release of Tiki 21.1</td>
</tr>
<tr>
<td>21.x LTS</td>
<td>2020-03</td>
<td>at least until 2022-03</td>
<td>2025-03</td>
</tr>
<tr>
<td>22.x</td>
<td>2020-11</td>
<td>until End of Life</td>
<td>not before the release of Tiki 23.1</td>
</tr>
<tr>
<td>23.x</td>
<td>2021-08</td>
<td>until End of Life</td>
<td>not before the release of Tiki 24.1</td>
</tr>
<tr>
<td>24.x LTS</td>
<td>2022-03</td>
<td>at least until 2024-03</td>
<td>2027-03</td>
</tr>
<tr>
<td>25.x</td>
<td>2022-12</td>
<td>until End of Life</td>
<td>not before the release of Tiki 26.1</td>
</tr>
<tr>
<td>26.x</td>
<td>2023-06 (planned)</td>
<td>until End of Life</td>
<td>not before the release of Tiki 27.1</td>
</tr>
<tr>
<td>27.x LTS</td>
<td>2024-02 (planned)</td>
<td>at least until 2026-02</td>
<td>2029-02</td>
</tr>
<tr>
<td>28.x</td>
<td>2024-10 (planned)</td>
<td>until End of Life</td>
<td>not before the release of Tiki 29.1</td>
</tr>
<tr>
<td>29.x</td>
<td>2025-06 (planned)</td>
<td>until End of Life</td>
<td>not before the release of Tiki 30.1</td>
</tr>
<tr>
<td>30.x LTS</td>
<td>2026-02 (planned)</td>
<td>at least until 2028-02</td>
<td>2031-02</td>
</tr>
</tbody>
</table>

- **D**: Development phase in trunk (About 6 months)
- **B**: Stabilization phase, from branch creation (forking of trunk) to initial release - about six weeks to 2 months before the release
- **R**: Regular support period with bug fixes and minor, self-contained enhancements (normally 8
months; 24 months for LTS versions)
- **S**: Security fixes only (36 months for LTS versions)
- **E**: End-of-Life (EoL)

A visual chart of the support period

This chart covers versions 18.x to 30.x with real data for the past, and projected data for the future (as of 2022-07). Please note that reality is slightly different than the plan (If one version is late, we will extend support period of other versions accordingly).

Examples

![Visualization of support periods for versions 18.x to 30.x]

### Strategies

Site admins have many choices.

<table>
<thead>
<tr>
<th>Main concern</th>
<th>Reason</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>I'll be developing new features</td>
<td>Tiki has most of what I need, but I also want to add X</td>
<td>Develop in trunk.</td>
</tr>
<tr>
<td></td>
<td>Your features will be in the stable/official Tiki version within months, and it will become community supported. Please see how to get commit access.</td>
<td></td>
</tr>
<tr>
<td>Main concern</td>
<td>Reason</td>
<td>Strategy</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>I want the new features</td>
<td>Because they are cool!</td>
<td><strong>Upgrade with each new stable version</strong>. Depending on your eagerness vs tolerance to bugs (capacity to fix them), you may want to skip x.0 versions.</td>
</tr>
<tr>
<td>I want the least possible number of bugs.</td>
<td>I have a huge user-base and they are particular.</td>
<td><strong>Go from one LTS to the next starting from midway in the series (e.g. starting from x.3 or x.4)</strong>. Since the version you are using will have been more in use, bugs will be rarer. You should at least test (and if possible deploy) before the security-only period so you may detect any bugs, report them (and ideally, you arrange to get them fixed).</td>
</tr>
<tr>
<td>I want to upgrade less often. Once I deploy, I want minimal changes.</td>
<td>A heavily customized theme or need to audit the code</td>
<td><strong>Use an LTS version, and just before it reaches end of life, upgrade to the latest LTS.</strong></td>
</tr>
<tr>
<td>I want the best possible security.</td>
<td>My site is a potential target.</td>
<td>This is a tricky one, and it depends on your use case (type of site, features, etc.). Could be: 1- <strong>Stay on latest stable version all the time</strong>, as it will have all the possible fixes and enhancements to security. 2- <strong>Move from one LTS to another (ex.: 9.x to 12.x) but start using when it is in security-fix-only phase</strong>. Since the version you are using will be over two years old, most vulnerabilities have been found and resolved. But you won’t have access to the latest security features. 3- Another option is to <strong>stay on latest LTS version all the time</strong>. You will have more security features, but you may experience more vulnerabilities (the ones that are introduced in one version but fixed in the next).</td>
</tr>
</tbody>
</table>

If you are using Tiki LTS, you should consider deploying on an LTS server, such as Ubuntu LTS. Debian and CentOS are also somewhat LTS in nature. Or go for WikiSuite and pick Virtualmin.

**Upgrade paths examples**

<table>
<thead>
<tr>
<th>Use case</th>
<th>Upgrade path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eager for new features</td>
<td>12.0 12.1 12.2 12.x... 13.0 13.1 13.2 13.x... 14.0 14.1 14.2 14.x... 15.0 15.1 15.2 15.x... 16.0 16.1 16.2 16.x...</td>
</tr>
<tr>
<td>Use case</td>
<td>Upgrade path</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>Prudent (skip x.0 releases)</td>
<td>12.1 12.2 12.x... 13.1 13.2 13.x... 14.1 14.2 14.x... 15.1 15.2 15.x... 16.1 16.2 16.x...</td>
</tr>
<tr>
<td>LTS</td>
<td>12.2 12.3 12.x... 15.2 15.3 15.x... 18.2 18.3 18.x...</td>
</tr>
</tbody>
</table>

**PHP versions**

Thanks to the availability of LTS versions, Tiki can be aggressive in increasing the requirements and take advantage of new possibilities of PHP. Indeed, if your infrastructure is not evolving as quickly (which is typical in large enterprise settings), you can use the Long Term Support (LTS) versions of Tiki. Please see: [Tiki Requirements](#).

**Browser support considerations**

For desktop browsers, Tiki generally supports the current and previous stable versions. Firefox and Chrome have a rapid release cycle with auto-update and, for Safari and Opera, users upgrade fairly quickly. So except for Internet Explorer and mobile browsers, we only have to worry about fairly recent browsers. For Internet Explorer, because upgrades are decided by the IT department or limited by the OS version, a significant portion of the user base doesn't use the latest version. Balancing the needs of innovation and stability, the Long Term Support (LTS) versions are perfect for Tiki to take advantage of all the new possibilities of newer browsers, while still offering a supported version for users that need to support older browsers.

<table>
<thead>
<tr>
<th>Tiki version</th>
<th>Internet Explorer and mobile browser version</th>
</tr>
</thead>
<tbody>
<tr>
<td>6LTS</td>
<td>IE6+</td>
</tr>
<tr>
<td>7, 8, 9LTS</td>
<td>IE7+</td>
</tr>
<tr>
<td>10, 11, 12LTS</td>
<td>IE8+, which is the highest you can upgrade to if you have Win XP</td>
</tr>
<tr>
<td>13, 14, 15LTS, 16</td>
<td>IE9+: see the discussion and stats. Note: Since IE11 has now been released, it is possible that support will be for IE10 and IE11 only, depending on how it goes with the Bootstrap integration.</td>
</tr>
<tr>
<td>17-18</td>
<td>IE 9+, Mobile Android 4+, Mobile Safari 7+</td>
</tr>
<tr>
<td>19+</td>
<td>IE 10 and below not officially supported. IE 10 note</td>
</tr>
</tbody>
</table>

**The past**

2009: Tiki moved to time-based releases. Regular releases every 6 months, and every third release was a Long Term Support (LTS) versions with a 2 year support period (starting with Tiki 3.x)
2012: The LTS support period was extended to 3.5 years (applicable to 6.x LTS and 9.x LTS)
2013: Extended once more, to 5 years (applicable for 12.x LTS and there-after)
2018: Changed to regular releases every 8 months.

**The Future**

As of 2018-11, the release cycle is not expected to change for the foreseeable future. However, if it does change, the same general principle will stay the same. And the total support period (as announced when you download) will not be reduced (but could be extended). Any eventual (and at this point, unlikely) change in policy will be amply discussed within the community before it happens. Potential changes are:
• **A faster cycle speed:** for example, every four months, in February, June, October, February, etc. or every six months in October and April. It could even be a combination (for example, 4 months, 4 months, and then 8 months). This would be while maintaining LTS versions for approximately the same periods. So instead of an LTS every third version (18 months), it would be every fourth (16 months) or fifth version (20 months).

• **Alignment with larger ecosystem:** Ubuntu’s Mark Shuttleworth once wrote: "There’s one thing that could convince me to change the date of the next Ubuntu LTS: the opportunity to collaborate with the other, large distributions on a coordinated major / minor release cycle.". The Tiki community would certainly participate in any similar initiative.

Related links

• http://php.net/supported-versions.php
• Composer 2.2 is LTS
• https://blogs.oracle.com/javamagazine/post/java-long-term-support-lts
• Where to commit