

## Automatic updates

This is the recipe for 1-click updates/upgrades starting in [Tiki25](#).

- Updates are in the same branch: 24.1 -> 24.2
- Upgrades are different branches: 21.3 -> 22.0 or 21.3 -> 24.1

## Context

1-click updates/upgrades is a highly-requested capability which is tricky to do right. Risks include:

- broken updates/upgrades
- time-outs
- browser disconnection (user closes browser during an upgrade or Internet access is lost)
- loss of local modifications
- lower security on file permissions
- issues if self-update conflicts with external installer tool
- The upgrade succeeded but the user wants to restore previous version (regression bug, change of behavior, etc.)

## Requirements

- Command line access
- Cron jobs
- Tiki Manager dependencies, or the capability to install them. See the Tiki Manager section of [Tiki Check](#)
  - rsync
  - git
  - etc.

## High-level

Once set up, Tiki will update Tiki Manager, and Tiki Manager will update/upgrade Tiki. Steps:

1. Install Tiki from Git, either
  - manually or
  - with an external [Tiki Manager](#) (which you don't need to keep after) or
  - with [virtualmin-tikimanager](#)
2. Set up the cron job for [Tiki Scheduler](#) (which will be used so operations are background processes)
3. Install the [Tiki Manager Package](#)
4. Use Tiki Manager to update/upgrade Tiki on demand (web or command line) or update automatically (via a cron job)

## Benefits

While tools have been available for years for updates/upgrades, they required using the command line. Starting in Tiki25, the command line will only be required for the initial setup.

## Code directly from Git sources

- Can get any revision, and not just released versions
- Can efficiently maintain local modifications
  - Before updates/upgrades, there is a check to detect conflicts
- Can use your own Git branch
- Can use merge requests / branches
- If a file is removed from the official source code, it will be removed. If you install over an older via a zip, you will have leftover files which can cause issues.

## Pick your preferred lifecycle

- Pick any version: Bleeding edge, stable, Long Term Versions (LTS), etc. See: [Versions](#)

## Sensible file permissions

Since Tiki Manager is not ran by the web user (like Apache or www), it can set safer file permissions

## Background process

Since Tiki Manager is not ran by the web user (like Apache or www), it is not prone to time-out errors

## Testing on clones

- Thanks to Tiki Manager, you can clone, clone-and-update or clone-and-upgrade to have a test environment. Within Virtualmin, this is even easier with the GUI to create a web space with a database, and clone from one to another, as seen here: <https://wikisuite.org/Virtualmin-Tiki-Manager>

## A Tiki instance can be managed by more than one Tiki Manager

- So for example, a hosting company could handle updates for security issues via a remote Tiki Manager, while the site manager uses the local Tiki Manager for other updates/upgrades like testing new features.

## Tons of other features

- Please see [Tiki Manager](#)

## Before Tiki25 via command line

## Tiki Manager

Recent versions of [Tiki Manager](#) use Git, but previously, it was SVN

- [instance:update](#)
- [instance:upgrade](#)
- [manager:setup-update](#)

## svnup.php

[svnup.php](#) which is bundled starting in [Tiki17](#)

**An example of this script on a daily cron job**

```
0 0 * * * cd /var/www/html/; php doc/devtools/svnup.php
```