Tiki Scheduler : automatisez avec une seule tâche cron

Tiki Scheduler : automatisez avec une seule tâche cron

 Bernard Sfez - 2026-07-02 21:09


La plupart des tâches de maintenance de Tiki Wiki — qu'il s'agisse de reconstruire l'index de recherche ou d'envoyer les résumés de notifications — ne devraient jamais dépendre de quelqu'un qui se souvient de cliquer sur un bouton. Le Scheduler intégré transforme ces corvées en tâches de fond : ajoutez une seule entrée cron sur votre serveur, puis gérez tout le reste depuis l'interface de Tiki.

La documentation officielle couvre les bases en quelques lignes ; ce guide va plus loin, en expliquant le raisonnement derrière chaque réglage, avec une liste de tâches que nous déployons réellement chez nos clients.

## Pourquoi le Scheduler est important dans Tiki

Un wiki, une plateforme collaborative ou un CMS n'est jamais vraiment « terminé » une fois installé. L'index de recherche se désynchronise à mesure que le contenu évolue, les résumés de notifications doivent partir selon un calendrier précis, les files d'attente d'e-mails doivent être vidées, et les vérifications de version doivent s'exécuter discrètement en arrière-plan. Avant que Tiki n'ait son propre Scheduler, chacune de ces tâches nécessitait sa propre ligne dans le crontab système, appelant son propre script PHP, selon son propre calendrier ; multipliez cela par une douzaine de tâches, et chaque migration vers un nouveau serveur impliquait de recopier une liste fragile de lignes cron.

La fonctionnalité Scheduler interne, introduite dans Tiki17, a résolu ce problème simplement : une seule entrée doit désormais vivre dans le crontab système. Cette entrée unique exécute `console.php scheduler:run`, qui fait partie de l'outil en ligne de commande Console intégré à Tiki, à intervalle fixe, et c'est Tiki lui-même qui détermine quelles tâches sont dues, les exécute et journalise le résultat. Tout le reste — ajouter une tâche, modifier son calendrier, la mettre en pause, consulter son historique — se fait depuis l'interface d'administration, et voyage avec votre instance Tiki au fil des sauvegardes et des changements de serveur.

Pour les équipes soucieuses de souveraineté numérique et d'opérations prévisibles, ce point compte : votre logique d'automatisation vit dans votre propre base de données Tiki, et non

éparpillée sur un serveur que vous ne maîtrisez pas forcément entièrement.

## Comment initier le Scheduler (tâche cron)

Vous le trouverez dans le module Scheduler Admin, sous « Settings » dans le menu principal de l'application, à l'adresse :
https://example.org/tiki-admin_schedulers.php
Depuis cette page, vous pouvez créer, lister et modifier des tâches. Vous avez également accès aux journaux, à leur dernier statut d'exécution, etc. Chaque tâche possède un Run Time exprimé en syntaxe cron standard — minute, heure, jour du mois, mois, jour de la semaine —, le même format que celui utilisé par les crontabs système sous GNU/Linux :

```
| * | * | * | * | *
| | | | +---- Jour de la semaine (plage : 1-7, 1 = lundi)
| | | +------ Mois (plage : 1-12)
| | +-------- Jour du mois (plage : 1-31)
| +---------- Heure (plage : 0-23)
+------------ Minute (plage : 0-59)
```

Ainsi, */5 * * * * signifie « toutes les 5 minutes », 0 1 * * * signifie « tous les jours à 1h du matin ». Si la syntaxe cron ne vous est pas familière, crontab.guru est un outil pratique pour vérifier une expression avant de l'enregistrer.

## Configurer la tâche cron principale

La tâche cron principale n'a qu'un seul rôle : demander à Tiki, fréquemment, « y a-t-il quelque chose à faire ? ». Elle doit s'exécuter au moins aussi souvent que votre tâche la plus fréquente ; une fois par minute est courant, une fois toutes les 5 ou 15 minutes suffit pour la plupart des tâches de maintenance. Cela se fait en envoyant une commande à la console Tiki, qui se trouve dans votre répertoire Tiki principal.

Commencez par ajouter ceci au crontab de votre serveur (adaptez le chemin vers la racine de votre Tiki) :

Pour l'exécuter toutes les 15 minutes
*/15 * * * * php /votre/chemin/tiki/console.php scheduler:run

Pour l'exécuter toutes les heures
0 * * * * /votre/chemin/tiki/console.php scheduler:run

La propriété du processus est importante. Si l'une de vos tâches planifiées écrit des fichiers de cache, ce qui est le cas de la plupart des tâches de maintenance, la tâche cron doit s'exécuter avec le même utilisateur que votre serveur web (www-data, apache, nginx…), et non root. L'exécuter avec le mauvais utilisateur est l'une des causes les plus fréquentes du « ça fonctionne quand je l'exécute manuellement, mais pas depuis cron ».

Si vous gérez plusieurs instances Tiki, Tiki Manager peut configurer cela automatiquement pour vous (instance:setup-scheduler-cron), ce qui vaut la peine d'être connu si vous maintenez plus

d'un ou deux sites, ou si vous faites appel à nos services de support Tiki Wiki pour garder plusieurs instances clientes synchronisées.
scheduler:run n'est pas la seule commande liée au Scheduler à connaître. Trois autres commandes vivent dans le même espace de noms scheduler et sont utiles pour le dépannage en ligne de commande plutôt que depuis l'interface :

- `php console.php scheduler:stats` # tableau des statistiques des tâches du Scheduler
- `php console.php scheduler:monitor` # surveille les tâches du Scheduler
- `php console.php scheduler:heal` # répare les tâches planifiées bloquées dans un état incorrect

`scheduler:heal` mérite particulièrement d'être retenue : si une tâche reste bloquée en affichant « running » après un redémarrage du serveur ou un plantage du processus PHP, c'est la commande qui la débloque sans avoir à toucher directement à la base de données.

### Maîtriser la taille de vos journaux

C'est un détail que la page officielle ne mentionne pas, et c'est la première chose à ajuster après avoir activé le Scheduler. Chaque exécution de chaque tâche écrit une entrée de journal, et par défaut, Tiki est généreux : l'historique peut atteindre 10 000 entrées par tâche. Sur une instance active avec plusieurs tâches s'exécutant toutes les quelques minutes, cette table de journaux se remplit rapidement et alourdit inutilement votre base de données pour un bénéfice pratique très limité ; personne ne lit dix mille lignes de « index:rebuild completed successfully ».

Rendez-vous sur tiki-admin.php?page=general#col1 et recherchez « Number of logs to keep » (ou utilisez le champ de recherche du panneau d'administration) pour ajuster la préférence scheduler_keep_logs. Pour la plupart des sites, 300 à 500 entrées suffisent largement, offrant confortablement quelques semaines à quelques mois d'historique selon la fréquence de vos tâches, tout en gardant la table de journaux légère.

### Tester le Scheduler : index:rebuild est votre meilleur allié

Une fois la tâche cron principale en place, n'attendez pas qu'une vraie tâche arrive à son échéance pour vérifier que tout fonctionne. La tâche la plus simple, et sans doute la plus utile pour tester, est la reconstruction de l'index de recherche unifié. Elle est incluse comme modèle (désactivé) par défaut depuis Tiki29 : « Rebuild index every day ». Ajustez votre cron principal et le cron de la tâche du Scheduler pour qu'elle s'exécute dans les 5 prochaines minutes environ. Observez le statut de la tâche dans le panneau du Scheduler passer à « success » après la prochaine exécution du cron principal. Une fois la vérification faite, remettez un calendrier raisonnable, une fois par nuit est typique pour la plupart des sites.

Ce test unique vous confirme trois choses à la fois : la tâche cron principale se déclenche, Tiki repère correctement les tâches dues, et la propriété du processus est correcte, puisqu'une reconstruction d'index échouera bruyamment si les répertoires de cache/index ne sont pas accessibles en écriture pour l'utilisateur du cron.

## Des tâches utiles à planifier

Une fois le mécanisme éprouvé, le Scheduler devient l'endroit naturel pour automatiser tout ce que vous exécuteriez sinon à la main. Voici quelques tâches que nous mettons en place

systématiquement, toutes via une ConsoleCommandTask avec une commande console.php :

- Garder votre sitemap XML à jour, pour que les moteurs de recherche voient toujours le contenu actuel : `sitemap:generate`
- Importer automatiquement des fichiers déposés dans un dossier du serveur vers une galerie de fichiers, pratique pour des exports automatisés depuis un autre système, des scanners, ou des scripts déposant des fichiers via SFTP : `php console.php files:batchupload 1 --subdirToSubgal --createSubgals` (ici, 1 est l'ID de la galerie cible ; ajoutez --fileUser www-data --fileGroup www-data si les permissions doivent être ajustées après l'import, et --filesPath /var/www/other/uploads si le dossier de dépôt se trouve en dehors du répertoire de batch par défaut.)
- Optimiser l'index de recherche sans reconstruction complète, plus léger que `index:rebuild`, utile entre deux reconstructions complètes sur les sites plus volumineux : `index:optimize`
- Nettoyer les jetons de connexion expirés et les caches obsolètes chaque nuit : `tokens:clear`, `cache:clear`
- Effectuer une sauvegarde planifiée des fichiers (`backup:files` requiert un argument de chemin de destination) : `php console.php backup:files /home/destination/path/backups/files`

C'est une bonne habitude en général avec console.php : exécutez toute nouvelle commande à la main une première fois, avant de l'intégrer dans une tâche planifiée. C'est le moyen le plus rapide de repérer un argument manquant ou un problème de permissions pendant que vous observez, plutôt qu'une semaine plus tard dans une ligne de journal silencieuse.

Chacune de ces tâches ne représente qu'un ajout d'une ligne une fois la tâche cron principale en place. Comme évoqué plus haut, Tiki peut aussi envoyer automatiquement un e-mail à un administrateur si une tâche planifiée échoue, puis à nouveau une fois qu'elle est rétablie, configurable via les préférences scheduler_users_to_notify_on_stalled et scheduler_users_to_notify_on_healed.

## Aller plus loin : (presque) aucune limite à ce que vous pouvez automatiser

Les exemples ci-dessus utilisent tous ConsoleCommandTask, mais ce n'est pas le seul type de Task pris en charge par le Scheduler : TikiCheckerCommandTask (utilisé par la tâche prédéfinie « Tiki Check ») en est un autre, intégré nativement, et le Scheduler est conçu pour être extensible au-delà des commandes console.php.

En particulier, dans les préférences avancées de Sécurité, il existe une option qui permet à une tâche planifiée d'exécuter directement une commande shell brute, sans passer par console.php. Vérifiez tiki-admin.php?page=security sur votre propre instance (filtrez les préférences avec « scheduler » ou « shell ») pour confirmer le libellé exact et l'état par défaut selon votre version, car c'est exactement le type de réglage qui devrait rester désactivé sauf besoin spécifique. Une fois activée, une tâche planifiée devient en pratique une tâche cron qui vit dans l'interface même de Tiki, avec journalisation et interface utilisateur, plutôt qu'une ligne brute dans /etc/crontab.

Nous préférons rester prudents ici : n'activez les tâches à commande shell que sur les instances que vous administrez entièrement, gardez restreinte la liste des personnes pouvant créer des tâches du Scheduler (permission tiki_p_admin_schedulers), et privilégiez les commandes console.php standard chaque fois qu'elles font déjà le travail. La liste complète et à jour des commandes disponibles vaut toujours la peine d'être consultée dans la documentation Console,

car de nouvelles commandes sont ajoutées à presque chaque version.

Pour donner une idée des posibilités, voici une liste non exhaustive de ce qu'il est réaliste d'automatiser sur une instance Tiki standard, sans code personnalisé, construite directement à partir de la liste actuelle des commandes console.php :

| **Tâche** | **Comment la configurer (commande console.php)** | **Remarques** |
|---|---|---|
| Reconstruire l'index de recherche | index:rebuild | La plus lourde mais la plus fiable ; bon choix par défaut chaque nuit |
| Optimiser l'index de recherche | index:optimize | Alternative plus légère entre deux reconstructions complètes |
| Générer le sitemap XML | sitemap:generate | Garde les moteurs de recherche synchronisés avec le nouveau contenu |
| Envoyer les résumés de notifications | notification:digest | Nécessite que les utilisateurs soient abonnés au mode résumé |
| Envoyer les rapports quotidiens des utilisateurs | daily-report:send | Fonctionnalité Daily Reports de Tiki |
| Vider la file d'attente d'e-mails | mail-queue:send | Pertinent uniquement si l'envoi des e-mails sortants est réglé sur « Queue » |
| Importer par lot les fichiers déposés | files:batchupload 1 --subdirToSubgal --createSubgals | Idéal pour des exports automatisés arrivant via SFTP |
| Vider les caches de Tiki | cache:clear | Sans risque à exécuter quotidiennement ; se régénère au chargement suivant |
| Supprimer les jetons expirés | tokens:clear | Nettoyage interne, sans effet visible pour les utilisateurs |
| Supprimer les cookies de connexion expirés | users:remove-cookies | Complète tokens:clear |
| Recalculer les champs mathématiques des trackers | tracker:recalc | Utile seulement si des trackers utilisent des champs calculés |
| Actualiser les flux RSS entrants | rss:refresh | Utile si vous agrégez des flux externes |
| Supprimer les anciens journaux système | log:delete | S'associe bien à une politique de rétention définie |

| Tâche | Comment la configurer (commande console.php) | Remarques |
|---|---|---|
| Synchroniser les abonnements de calendrier | calendar:sync | Pour les calendriers récupérant des flux ICS externes |
| Mettre à jour Tiki via le contrôle de version | vcs:update | Réservé de préférence à un environnement de staging, pas en production sans surveillance |
| Sauvegarder les fichiers de l'instance | backup:files /path/to/backup | Nécessite un chemin de destination accessible en écriture, voir plus haut |
| Sauvegarder la base de données | database:backup /path/to/backup | À combiner avec backup:files pour une sauvegarde nocturne complète |
| Vérifier les paquets installés | package:update | À examiner avant toute activation automatique en production |

Comme avec `backup:files`, certaines de ces commandes (`database:backup`, `files:batchupload` attendent un chemin ou un ID de galerie en argument plutôt que de s'exécuter seules ; le principe vu plus haut s'applique : testez la commande exacte à la main une première fois, puis intégrez-la dans une tâche planifiée avec cette même syntaxe fonctionnelle.

## Pour conclure

Le Scheduler fait partie de ces fonctionnalités de Tiki qui se rentabilisent discrètement : quelques minutes de configuration, et votre index de recherche, vos notifications, votre sitemap et vos tâches de maintenance s'exécutent tout seuls, de manière fiable, avec un historique visible que vous pouvez consulter d'un coup d'œil. Nous espérons que cela vous donne une vision plus complète que la page de référence, ainsi que quelques commandes prêtes à l'emploi pour démarrer.

Si vous préférez que tout cela soit configuré, surveillé et maintenu à jour pour vous, c'est exactement à cela que servent nos services de support Tiki Wiki. Nous sommes toujours ravis de vous aider.

## Pour aller plus loin

- Documentation Tiki : Scheduler, Cron, Console, Cron Job to Rebuild Search Index, Manager
- Profil : Scheduler_presets_20