Tiki Manager

Tiki Manager permits you to manage various instances of Tiki. You can install, upgrade, backup, clone, check the file integrity and do various other things. This can be done on the same server or a remote server. Most actions can be run unattended on a cron job. It can be used via the command line or a web interface.

Tiki Manager supercedes TRIM.

Screenshot of the Tiki Manager Web interface

More info and screenshots at Tiki Manager Web UI.

Requirements

- SQLite3 for data storage
- PHP 7.1+ with Command-line access (CLI). It uses the Symfony Console Component. It uses Composer for dependency management.
- Git
- Standard server tools like rsync, unzip, etc.
- It’s been developed on GNU/Linux, and designed to work anywhere PHP, SQLite and Git are available (Windows, MacOS, *BSD, etc.)

You can use Check to verify that your server can run Tiki or Tiki Manager
Installation

For now, the installation instructions are made for ClearOS so you need to adapt for your OS. Please see How to install Tiki Manager on ClearOS. See also Additional configuration below.

At a high level:

1. Use Server Check to review and get all the server dependencies
2. Get code from Git in a non-web accessible directory: https://gitlab.com/tikiwiki/tiki-manager.git
3. Run Tiki Manager with `php tiki-manager.php` to complete set up. It will instruct you how to do so (ex.: run Composer to get PHP dependencies)

Example commands (that worked for me on Debian 9 )

```
# Create a directory called tiki-manager (you can pick another name)
$ mkdir tiki-manager
# Move into this new directory
$ cd tiki-manager
# Fetch Tiki Manager directly from sources. The last period/dot (.) indicates to put files in current directory (vs creating a new directory)
$ git clone --depth 1 --branch master https://gitlab.com/tikiwiki/tiki-manager.git .
# Then run the following. On first run, it will finish the setup (Fetch Composer and get PHP dependencies, Create the SQLite database, etc.)
$ php tiki-manager.php manager:info
```

Documentation

Here is a brief explanation of the basic commands that can be used. All commands follow a wizard pattern.

For more help, you can add --help to any command as in this example

```
php tiki-manager.php instance:create --help
```

instance

instance:create

Adds an instance to be managed by Tiki Manager, which can mean either:

- create a fresh Tiki instance (including a database) fetching files from Git (and optionally SVN)
- or detect an existing Tiki installed via Git or Subversion (SVN) and "adopt" it.
- You may need to convert your Tiki instance from FTP to Git/SVN.

<table>
<thead>
<tr>
<th>Params</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>blank</td>
<td>option</td>
<td>Blank Instance</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>Instance connection type</td>
<td></td>
</tr>
<tr>
<td>host</td>
<td>Remote host name</td>
<td></td>
</tr>
<tr>
<td>port</td>
<td>Remote port number</td>
<td></td>
</tr>
<tr>
<td>user</td>
<td>Remote User</td>
<td></td>
</tr>
<tr>
<td>pass</td>
<td>Remote password</td>
<td></td>
</tr>
<tr>
<td>url</td>
<td>Instance web URL</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>Instance name</td>
<td></td>
</tr>
<tr>
<td>email</td>
<td>Instance contact email</td>
<td></td>
</tr>
<tr>
<td>webroot</td>
<td>Instance web root</td>
<td></td>
</tr>
<tr>
<td>tempdir</td>
<td>Instance temporary directory</td>
<td></td>
</tr>
<tr>
<td>branch</td>
<td>Instance branch</td>
<td></td>
</tr>
<tr>
<td>backup-user</td>
<td>Instance backup user</td>
<td></td>
</tr>
<tr>
<td>backup-group</td>
<td>Instance backup group</td>
<td></td>
</tr>
<tr>
<td>backup-permission</td>
<td>Instance backup permission</td>
<td></td>
</tr>
<tr>
<td>db-host</td>
<td>Instance database host</td>
<td></td>
</tr>
<tr>
<td>db-user</td>
<td>Instance database user</td>
<td></td>
</tr>
<tr>
<td>db-pass</td>
<td>Instance database password</td>
<td></td>
</tr>
<tr>
<td>db-prefix</td>
<td>Instance database prefix (creates an user and database)</td>
<td></td>
</tr>
<tr>
<td>db-name</td>
<td>Instance database name (when db-prefix is not used)</td>
<td></td>
</tr>
</tbody>
</table>

```shell
php tiki-manager.php instance:create
```

Non interactive command:

```shell
php tiki-manager.php instance:create --type=local --url=http://manager.tiki.org --name=manager.tiki.org --email=manager@example.org --webroot=/www/manager --tempdir=/tmp/trim_temp --branch=branches/19.x --backup-user=www-data --backup-group=www-data --backup-permission=750 --db-host=localhost --db-user=root --db-pass=secret --db-prefix=manager
```

```shell
php tiki-manager.php instance:list
```

### instance:list

```shell
php tiki-manager.php instance:list
```

<table>
<thead>
<tr>
<th>Params</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>json</td>
<td>option</td>
<td>Instance list in a parsable format</td>
</tr>
</tbody>
</table>

### instance:access

Opens a shell to the remote host.

Since Tiki Manager manages all your connections, using this command simply avoids needing to remember...
passwords.

**Params**

<table>
<thead>
<tr>
<th>Params</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>instances</td>
<td>option</td>
<td>List of instance IDs or Names to be checked, separated by comma (,)</td>
</tr>
</tbody>
</table>

```php
tiki-manager.php instance:access
```

**instance:backup**

Tiki Manager performs a complete backup of the Tiki instance. The backup includes all files (using rsync for efficient bandwidth management) on remote host (including user files and files stored out of the web root) and a dump of the database. Each backup is archived in the backup/archive folder as a .tar.bz2 file, where Tiki Manager is installed.

The backup file contains:

- `manifest.txt`: Indicates where folders were located on the remote host
- `database_dump.sql`: Self-explanatory
- `[a-f0-9]{32}`: Folders named using a hash. Content of the folders on remote host. The `manifest.txt` file lists these hashes.

**Params**

<table>
<thead>
<tr>
<th>Params</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>instances</td>
<td>option</td>
<td>List of instance IDs or Names to be checked, separated by comma (,)</td>
</tr>
<tr>
<td>exclude</td>
<td>option</td>
<td>Used with --instances=all, a list of instance IDs to exclude from backup</td>
</tr>
<tr>
<td>email</td>
<td>option</td>
<td>Email addresses to notify for backup failures (comma separated)</td>
</tr>
<tr>
<td>max-backups</td>
<td>option</td>
<td>Max number of backups to keep by instance</td>
</tr>
</tbody>
</table>

```php
tiki-manager.php instance:backup
```

**instance:blank**

Like instance:create but it doesn't actually add a Tiki. It just creates an instance that can be used to restore or clone another instance.

```php
tiki-manager.php instance:blank
```
instance:check

This is equivalent to a secdb check, except that it's safer because it's made from a trusted host and will take your custom modifications on host into consideration. Upon first run, the check will ask where it should fetch the hashes from.

**Params**

<table>
<thead>
<tr>
<th>Params</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>instances</td>
<td>option</td>
<td>List of instance IDs or Names to be checked, separated by comma (,)</td>
</tr>
<tr>
<td>update-from</td>
<td>option</td>
<td>Action related to how checksums are performed. Accepted values - current or source.</td>
</tr>
</tbody>
</table>

```php
tiki-manager.php instance:check
```

instance:clone

Makes another identical copy of Tiki. This is basically a combination of make backup and make restore in one operation.

The destination instance can be blank or another configured and managed Tiki Instance. Be very careful, when using existing another Tiki Instance, as the data here will be wiped and replaced by the data from the source instance.

As two Tiki instances with the same settings (ex.: both point to the same Elasticsearch index can cause issues), recommended reading: [Divergent Preferences in Staging Development Production](https://example.com)

The following are excluded from the cloning process:

- `db/local.php` (the database access info needs to point to another database)
- the `.ini.php` file (if any) used for System Configuration

**Params**

<table>
<thead>
<tr>
<th>Params</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>parameter</td>
<td>Check if is a clone or upgrade</td>
</tr>
<tr>
<td>check</td>
<td>option</td>
<td>Check files checksum. Only used in mode upgrade.</td>
</tr>
<tr>
<td>skip-reindex</td>
<td>option</td>
<td>Skip rebuilding index step.</td>
</tr>
<tr>
<td>skip-cache-warmup</td>
<td>option</td>
<td>Skip generating cache step.</td>
</tr>
<tr>
<td>live-reindex</td>
<td>option</td>
<td>Set instance maintenance off and after perform index rebuild.</td>
</tr>
<tr>
<td>direct</td>
<td>option</td>
<td>Use rsync to copy files between local instances.</td>
</tr>
<tr>
<td>keep-backup</td>
<td>option</td>
<td>Source instance backup is not deleted before the process finished.</td>
</tr>
<tr>
<td>use-last-backup</td>
<td>option</td>
<td>Use source instance last created backup.</td>
</tr>
<tr>
<td>db-host</td>
<td>option</td>
<td>Target instance database host</td>
</tr>
<tr>
<td>db-user</td>
<td>option</td>
<td>Target instance database user</td>
</tr>
<tr>
<td>db-pass</td>
<td>option</td>
<td>Target instance database password</td>
</tr>
</tbody>
</table>
instance:cloneandupgrade

Like instance:clone but with an extra upgrade operation.

As two Tiki instances with the same settings (ex.: both point to the same Elasticsearch index can cause issues), recommended reading: Divergent Preferences in Staging Development Production

The following are excluded from the cloning process:

- db/local.php (the database access info needs to point to another database)
- the .ini.php file (if any) used for System Configuration

### Params

<table>
<thead>
<tr>
<th>Params</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>parameter</td>
<td>Check if is a clone or upgrade.</td>
</tr>
<tr>
<td>check</td>
<td>option</td>
<td>Check files checksum. Only used in mode upgrade.</td>
</tr>
<tr>
<td>skip-reindex</td>
<td>option</td>
<td>Skip rebuilding index step.</td>
</tr>
<tr>
<td>skip-cache-warmup</td>
<td>option</td>
<td>Skip generating cache step.</td>
</tr>
<tr>
<td>live-reindex</td>
<td>option</td>
<td>Set instance maintenance off and after perform index rebuild.</td>
</tr>
<tr>
<td>direct</td>
<td>option</td>
<td>Use rsync to copy files between local instances.</td>
</tr>
<tr>
<td>keep-backup</td>
<td>option</td>
<td>Source instance backup is not deleted before the process finished.</td>
</tr>
<tr>
<td>use-last-backup</td>
<td>option</td>
<td>Use source instance last created backup.db-host</td>
</tr>
<tr>
<td>source</td>
<td>option</td>
<td>Use a certain source instance, ID or name.</td>
</tr>
<tr>
<td>target</td>
<td>option</td>
<td>Use a certain target instance, ID or name.</td>
</tr>
<tr>
<td>branch</td>
<td>option</td>
<td>The branch used for the upgrade.</td>
</tr>
<tr>
<td>db-user</td>
<td>option</td>
<td>Target instance database user.</td>
</tr>
<tr>
<td>db-pass</td>
<td>option</td>
<td>Target instance database password.</td>
</tr>
<tr>
<td>db-prefix</td>
<td>option</td>
<td>Target instance database prefix (creates an user and database).</td>
</tr>
<tr>
<td>db-name</td>
<td>option</td>
<td>Target instance database name (when db-prefix is not used).</td>
</tr>
</tbody>
</table>

php tiki-manager.php instance:cloneandupgrade
instance:console
Allow to run any Console command from Tiki.

```
php tiki-manager.php instance:console
```

instance:copysshkey
Copy the SSH key to the remote instance.

```
php tiki-manager.php instance:copysshkey
```

instance:delete
Delete the instance via the command line (you could also do via the web interface). This does NOT delete your Tiki. It just deletes the instance connection to it.

```
php tiki-manager.php instance:delete
```

instance:detect
Detect Tiki branch or tag, and PHP version. For debugging purposes. Also useful if you manually proceeded to svn switch and Tiki Manager needs to update its internal database about a Tiki instance.

```
php tiki-manager.php instance:detect
```

instance:edit
Permits to modify an instance.

```
php tiki-manager.php instance:edit
```

instance:fixpermissions
Run setup.sh on the remote host using automated parameters. It should work in most cases. If the command proposed my setup.sh without parameters or super user rights are required, you should connect to the remote host manually using `instance:access`.

```
php tiki-manager.php instance:fixpermissions
```

instance:import
Import an instance to the instances list, if detects a Tiki instance, and it’s not yet managed by Tiki Manager.

**Params**

<table>
<thead>
<tr>
<th>Params</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>


<table>
<thead>
<tr>
<th>option</th>
<th>Instance connection type</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Instance connection type</td>
</tr>
<tr>
<td>host</td>
<td>Remote host name</td>
</tr>
<tr>
<td>port</td>
<td>Remote port number</td>
</tr>
<tr>
<td>user</td>
<td>Remote User</td>
</tr>
<tr>
<td>pass</td>
<td>Remote password</td>
</tr>
<tr>
<td>url</td>
<td>Instance web URL</td>
</tr>
<tr>
<td>name</td>
<td>Instance name</td>
</tr>
<tr>
<td>email</td>
<td>Instance contact email</td>
</tr>
<tr>
<td>webroot</td>
<td>Instance web root</td>
</tr>
<tr>
<td>tempdir</td>
<td>Instance temporary directory</td>
</tr>
</tbody>
</table>

```
php tiki-manager.php instance:import
```

**Non interactive command:**

```
php tiki-manager.php instance:import --type=local --url=http://manager.tiki.org --name=manager.tiki.org --email=manager@example.org --webroot=/www/manager --tempdir=/tmp/trim_temp
```

**instance:maintenance**

Put instances under maintenance or live mode.

```
php tiki-manager.php instance:maintenance
```

**instance:profile:apply**

Apply a profile to an instance.

```
php tiki-manager.php instance:profile:apply
```

**instance:restore**

Restore on a blank installation. Ref: instance:blank If you have data files that are not stored in the database, you should use the ideal scenario for data file storage and relative paths.

```
php tiki-manager.php instance:restore
```

**instance:setup-scheduler-cron**

This command allows you to enable the cron to run the schedulers

```
php tiki-manager.php instance:setup-scheduler-cron
```
instance:stats

Extract stats (KPIs) from selected instances as csv. Optionally that information can be pushed to another Tiki instance.

**Params**

<table>
<thead>
<tr>
<th>Params</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>instances</td>
<td>option</td>
<td>all or list instances to fetch KPI, separated by comma (,). Default is all</td>
</tr>
<tr>
<td>exclude</td>
<td>option</td>
<td>List of instance IDs to be excluded, separated by comma (,)</td>
</tr>
<tr>
<td>file</td>
<td>option</td>
<td>File name to output the stats. Required when --push-to is used.</td>
</tr>
<tr>
<td>push-to</td>
<td>option</td>
<td>Instance ID to push collected instance stats</td>
</tr>
</tbody>
</table>

```bash
# Display all instance stats
php tiki-manager.php instance:stats

# Save stats to a csv file
php tiki-manager.php instance:stats --instances=all --file=instance_stats.csv

# Upload instances stats to another tiki instance
# Using TIKI_ROOT will match instance webroot
php tiki-manager.php instance:stats --instances=all --exclude=1 --file=TIKI_ROOT/temp/instance_stats.csv --push-to=1
```

instance:update

- Does a dry-run first, and aborts on any conflicts
- Updates to latest code in that branch (or trunk) using svn up and thus merging any changes
- And all operations should be done after updating the code
  - Updates the file hashes accordingly. The hash verification/update may prompt with some files containing conflicts if modifications were made on the instance.
  - Performs the database update.
  - Runs setup and Composer,
  - Clears cache
  - Rebuilds search index

During the update process, the instance is disabled using a .htaccess file (previous one is preserved), making the site unavailable until the update is completed.

**Params**

<table>
<thead>
<tr>
<th>Params</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>parameter</td>
<td>Check if is an auto or switch mode</td>
</tr>
<tr>
<td>instances</td>
<td>option</td>
<td>List of instance IDs or Names to be checked, separated by comma (,)</td>
</tr>
<tr>
<td>branch</td>
<td>option</td>
<td>Instance branch to update</td>
</tr>
</tbody>
</table>
### instance:upgrade

Similar to update. Requests for the branch to switch to over the update. You will have several choices of branches. Please note that you should **NOT** downgrade as Tiki doesn't support a downgrade database script. An upgrade is a one-way street! You should make a backup before you upgrade so you can return to this version if issues arise.

If you choose to do a manual upgrade (with svn switch).

#### Params

<table>
<thead>
<tr>
<th>Param</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>check</td>
<td>option</td>
<td>Check files checksum.</td>
</tr>
<tr>
<td>skip-reindex</td>
<td>option</td>
<td>Skip rebuilding index step.</td>
</tr>
<tr>
<td>skip-cache-warmup</td>
<td>option</td>
<td>Skip generating cache step.</td>
</tr>
<tr>
<td>live-reindex</td>
<td>option</td>
<td>Set instance maintenance off and after perform index rebuild.</td>
</tr>
<tr>
<td>email</td>
<td>option</td>
<td>Email address to notify in case of failure. Use , (comma) to separate multiple email addresses.</td>
</tr>
<tr>
<td>lag</td>
<td>option</td>
<td>Time delay commits by X number of days. Useful for avoiding newly introduced bugs in automated updates.</td>
</tr>
</tbody>
</table>

### instance:verify

Verify an instance (same as check)

```
php tiki-manager.php instance:verify
```

### instance:watch

This command perform the Hash check (instance:check). The script will prompt for a contact email address to notify with the log in the case of a suspicious file change (which could be an intrusion or someone who legitimately changed a file on the server).
<table>
<thead>
<tr>
<th>Params</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>email</td>
<td>option</td>
<td>Email address to contact.</td>
</tr>
<tr>
<td>exclude</td>
<td>option</td>
<td>List of instance IDs to be excluded, separated by comma (,)</td>
</tr>
</tbody>
</table>

```php
tiki-manager.php instance:watch
```

### backups

### backups:setup

Same as `manager:setup-backups`.

```php
tiki-manager.php backups:setup
```

### backups:delete

Delete Tiki Manager backups folder and contents. This folder contains the backups of instances managed by Tiki Manager.

```php
tiki-manager.php backups:delete
```

### cache

### cache:clear

Delete Tiki Manager cache folder. Useful for development.

```php
tiki-manager.php cache:clear
```

### database

### database:delete

- Delete Tiki Manager database. Useful for development.

```php
tiki-manager.php database:delete
```

### database:view

- View Tiki Manager database. For debug purposes and useful for development.

```php
tiki-manager.php database:view
```
logs

logs:clear
Clear Tiki Manager logs folder.

```
php tiki-manager.php logs:clear
```

manager

manager:info

- Display running OS, PHP version and binary used by Tiki Manager

```
php tiki-manager.php manager:info
```

manager:check

- Check OS requirements to execute Tiki Manager

```
php tiki-manager.php manager:check
```

manager:report

- Reports, and send reports to a Tiki instance using Data Channels.

```
php tiki-manager.php manager:report
```

manager:reset

- Delete Tiki Manager backup, cache, and log files. Useful for development.

```
php tiki-manager.php manager:reset
```

manager:setup-backups

Set-up a cronjob to perform automatic instance(s) backups (instance:backup) every day at a specific time. The script will prompt for the time which the cron should run at and the instances that will be ignored by these automatic backups (all instances are selected by default).

- Automatic backups cronjob should not run at the same time as the "manager:setup-update" cron job command. Make sure you pick different run times.

**Params**

<table>
<thead>
<tr>
<th>Params</th>
<th>ParamsType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>option</td>
<td>Time to trigger the instance(s) backup using the format &lt;hours&gt;:&lt;minutes&gt;</td>
</tr>
</tbody>
</table>
manager:setup-backups

Set-up a cron job to perform automatic instance(s) update (instance:update) every day at a specific time. The script will prompt for the time which the cron should run at and the instances that will be affected by this automatic update.

**Params**

<table>
<thead>
<tr>
<th>Params</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>option</td>
<td>Time to trigger the instance(s) update using the format &lt;hours&gt;:&lt;minutes&gt;</td>
</tr>
<tr>
<td>instances</td>
<td>option</td>
<td>List of instance IDs or Names to be checked, separated by comma (,)</td>
</tr>
<tr>
<td>email</td>
<td>option</td>
<td>Email address to report update failures (multiple emails must be separated by comma (,))</td>
</tr>
</tbody>
</table>

manager:setup-watch

Set-up a cron job on the Tiki Manager master to perform the Hash check (instance:check) automatically every day. The script will prompt for a contact email address to notify with the log in the case of a suspicious file change (which could be an intrusion or someone who legitimately changed a file on the server) and the time at which the script should run.

**Params**

<table>
<thead>
<tr>
<th>Params</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>email</td>
<td>option</td>
<td>Email address to contact.</td>
</tr>
<tr>
<td>time</td>
<td>option</td>
<td>The time update should run.</td>
</tr>
<tr>
<td>exclude</td>
<td>option</td>
<td>List of instance IDs to be excluded, separated by comma (,)</td>
</tr>
</tbody>
</table>

manager:update

Update Tiki Manager to the latest version available. This updates the PHP code from Git, and updates the Composer dependencies. **As of 2021-07-19: This doesn't update the web interface. You need to**
redo the `webmanager:enable` operation below to make sure there is no mismatch between your CLI code and your webmanager code.

**Params**

<table>
<thead>
<tr>
<th>Params</th>
<th>Short</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--check</td>
<td>-c</td>
<td>option</td>
<td>Only checks if there is new version available for update.</td>
</tr>
<tr>
<td>--yes</td>
<td>-y</td>
<td>option</td>
<td>Say yes to update</td>
</tr>
</tbody>
</table>

```bash
code:php
php tiki-manager.php manager:update --check
```

```bash
code:php
php tiki-manager.php manager:update -y
```

For installations (without Phar or Git), to enable update, add the following file `.version` to the Tiki-Manager root.

```json
dict:json
{"version":"bf63ffa", "date":"2020-05-03T23:58+01:00"}
```

**webmanager**

**webmanager:enable**

Enable a web interface for Tiki Manager. More info at [Tiki Manager Web UI](#).

**Params**

<table>
<thead>
<tr>
<th>Params</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>option</td>
<td>Path to install webmanager</td>
</tr>
<tr>
<td>username</td>
<td>option</td>
<td>The username to login</td>
</tr>
<tr>
<td>password</td>
<td>option</td>
<td>The username's password</td>
</tr>
<tr>
<td>www-user</td>
<td>option</td>
<td>The apache user (set this if other than apache)</td>
</tr>
<tr>
<td>www-group</td>
<td>option</td>
<td>The apache group (set this if other than apache)</td>
</tr>
<tr>
<td>restrict</td>
<td>option</td>
<td>(No value) Restrict WebManager access to localhost</td>
</tr>
<tr>
<td>install</td>
<td>option</td>
<td>(No value) Proceed Tiki Manager installation (no prompt confirmation)</td>
</tr>
</tbody>
</table>

**Option 1: Interactive setup**

```bash
code:bash
php tiki-manager.php webmanager:enable
```

**Option 2: Non-Interactive setup**

```bash
code:bash
```
The Tiki Manager Web Interface will install its files as follows. Please note that in this example, the directory already contained `index.html` which must be deleted or renamed so `index.php` (installed for Tiki Manager’s web interface) can load up when visiting that URL.

Once set up and running, web interface will look like this

![Tiki Manager Web Administration](image)

webmanager:block

Webmanager block login
### Additional configuration

To easily configure the Tiki Manager application, copy `.env.dist` file to `.env` and insert your configurations for the uncommented (#) entries.

### Version Control System

Tiki Manager by default uses git and public repository. If you want to use SVN as your default vcs or another repository please use the following lines in your .env file.

```bash
DEFAULT_VCS=svn
GIT_TIKIWIKI_URI=<CUSTOM_GIT_REPOSITORY_URL>
SVN_TIKIWIKI_URI=<CUSTOM_SVN_REPOSITORY_URL>
```

### Behind proxy or without internet connection

Tiki Manager is able to use Tiki's distributed version packages as an alternative when there is no connection to external servers like gitlab or sourceforge.

Setting the default VCS to src, Tiki Manager will use existing packages in the data/tiki_src folder (default).

```bash
DEFAULT_VCS=src
```

Download the distributed Tiki packages, from [https://sourceforge.net/projects/tikiwiki/files/](https://sourceforge.net/projects/tikiwiki/files/), and save them into data/tiki_src folder.

### Email settings

To configure Tiki Manager email sender address add the following line to your .env file.

```bash
FROM_EMAIL_ADDRESS=<SENDER_EMAIL_ADDRESS>
```

### Configure SMTP Server

By default Tiki Manager uses sendmail to send email notifications. If you intend to use SMTP instead add the following lines to your .env file.

```bash
SMTP_HOST=<SERVER_ADDRESS>
SMTP_PORT=<SERVER_PORT>
```
Web Manager settings

If you want to setup a default folder to install your web manager or apache user:group are different than apache:apache you can add the following settings to your .env file.

| WWW_PATH=<WEB_MANAGER_FOLDER> |
| WWW_USER=<APACHE_USER>         |
| WWW_GROUP=<APACHE_GROUP>       |

To change the maximum number of failed login attempts on Web Manager, add the following setting to your .env file.

| MAX_FAILED_LOGIN_ATTEMPTS=<NUMBER> |

Timeouts during long running operations

During long operations (like clone or clone and upgrade) you may receive an HTTP error code 503 with a message "Service Unavailable", you can increase Apache's proxy timeout to a more suitable value.

To do that at the Virtual Host level (so it's only enabled for tiki manager) you need to do the following:

Add the Apache directive `ProxyTimeout` to Tiki Managers VirtualHost configuration file.
Example: `ProxyTimeout 300` will set the proxy timeout to 5 minutes

For ClearOS check the instructions: https://doc.tiki.org/How-to-install-Tiki-Manager-on-ClearOS#Timeouts_during_long_running_operations

Other Notes

Tiki Manager vs MultiTiki

Using the `instance:console` command in Tiki Manager you can access the multitiki commands in the remote instance, like this. Assuming we are using instance #42 and the virtual domain is example.com, and each branch is in the same directory (i.e. `tiki/branches/20.x` and `tiki/branches/21.x` in this instance.

Details

[+]

Source code

The source code is managed here: https://gitlab.com/tikiwiki/tiki-manager/
Roadmap

Related

- Tiki Manager app for ClearOS
- How to install Tiki Manager on ClearOS

alias

- Tiki Manager