

Realtime

New in [Tiki25](#). Use built-in websockets server within Tiki to enable realtime communication features of the app.

This is optional. If your server doesn't support this, it's OK. If you have it, supported features will be faster and smoother. For now, it is used for [Tiki Manager Package](#), and will later be added to other Tiki Features.

Powered by <https://packagist.org/packages/cboden/Ratchet> and <https://reactphp.org>

This will require a special server configuration (possibly root access). [tiki-check.php](#) will indicate if server is correctly set up.

In this documentation, we are going to configure Virtualmin (but any other server can be used), and then we will check if it worked.

CONFIGURATION

REQUIREMENTS

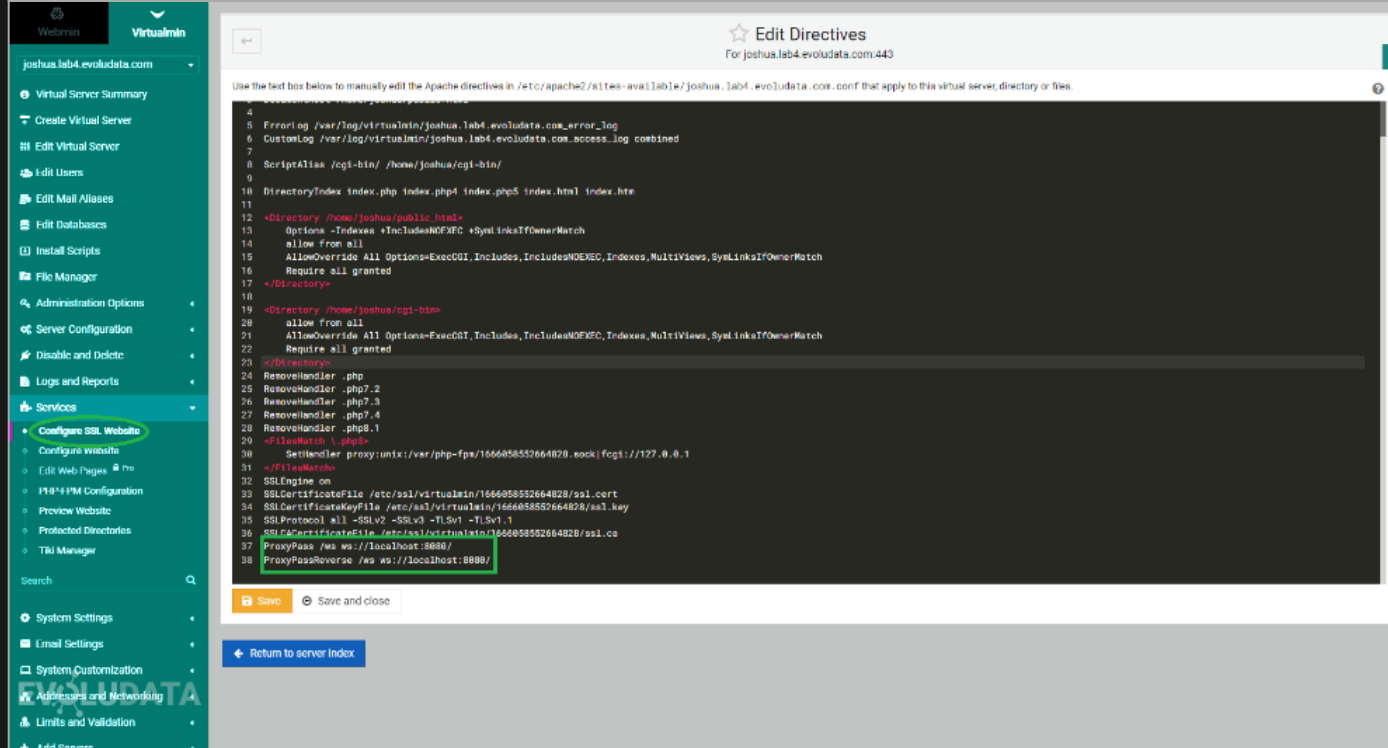
We already have a Tiki instance up and running, on Virtualmin, now we need to take into account a few more things: **Proxy Settings**, **Realtime Service**, **proxy_wstunnel**, **Realtime** feature, and **Realtime port** in Tiki in order to get Realtime working.

1. PROXY SETTINGS

First, let us edit Apache directives and add the ProxyPass and the ProxyPassReverse settings in the SSI virtual host. We make both point to `/ws ws://localhost:8080/`

To do this, we go to **Virtualmin** -> **Services** -> **Configure SSL Website** -> **Edit Directives**.

In the text box that opens, we add `ProxyPass /ws ws://localhost:8080/` and `ProxyPassReverse /ws ws://localhost:8080/` at the end, like shown in the picture below:



Click to expand

2. REALTIME SERVICE

At this step, we need to create a Systemd service to start the WS server.

I.B: Start the WS server with the same user that Tiki web requests run as (to avoid permission issues) - e.g. `sudo -u www-data php tiki-realtime.php`

The process is simple. Just go to **Webmin** -> **System** -> **Bootup and Shutdown** -> **create a new systemd service**.

The screenshot shows the Webmin interface for 'Bootup and Shutdown' on a Systemd boot system. A green circle highlights the 'Create a new systemd service' button. Below it is a table of existing services:

Service name	Service description	Service status	Start at boot?	Running now?
<input type="checkbox"/> apache-htcacheclean	Start the htcacheclean helper	Unknown	No	No
<input type="checkbox"/> apache-htcacheclean.service	Disk Cache Cleaning Daemon for Apache HTTP Server	Inactive (dead)	No	No
<input type="checkbox"/> apache2	Start the web server	Unknown	Yes	Yes
<input type="checkbox"/> apache2.service	The Apache HTTP Server	Active (running)	Yes	Yes
<input type="checkbox"/> apparmor	AppArmor init script. This script loads all AppArmor profiles.	Unknown	No	Unknown
<input type="checkbox"/> apparmor.service	Load AppArmor profiles	Active (exited)	Yes	Yes
<input type="checkbox"/> apt-daily-upgrade.service	Daily apt upgrade and clean activities	Inactive (dead)	Always	No
<input type="checkbox"/> apt-daily-upgrade.timer	Daily apt upgrade and clean activities	Active (waiting)	Yes	Yes
<input type="checkbox"/> apt-daily.service	Daily apt download activities	Inactive (dead)	Always	No
<input type="checkbox"/> basic.target	Basic System	Active (active)	Always	Yes
<input type="checkbox"/> boot-complete.target	Boot Completion Check	Inactive (dead)	Always	No
<input type="checkbox"/> certbot.service	Certbot	Inactive (dead)	Always	No
<input type="checkbox"/> certbot.timer	Run certbot twice daily	Active (waiting)	Yes	Yes
<input type="checkbox"/> console-getty.service	Console Getty	Inactive (dead)	No	No
<input type="checkbox"/> console-setup.service	Set console font and keymap	Active (exited)	Yes	Yes
<input type="checkbox"/> console-setup.sh	Set console font and keymap	Unknown	Yes	No
<input type="checkbox"/> cron	cron is a standard UNIX program that runs user-specified	Unknown	Yes	Yes
<input type="checkbox"/> cron.service	Regular background program processing daemon	Active (running)	Yes	Yes
<input type="checkbox"/> cryptdisks-early.service	cryptdisks-early.service	Inactive (dead)	No	No
<input type="checkbox"/> cryptdisks.service	cryptdisks.service	Inactive (dead)	No	No
<input type="checkbox"/> cryptsetup-pre.target	Local Encrypted Volumes (Pre)	Inactive (dead)	Always	No
<input type="checkbox"/> cryptsetup.target	Local Encrypted Volumes	Active (active)	Always	Yes
<input type="checkbox"/> dbus	D-Bus is a simple interprocess messaging system, used	Unknown	Yes	Yes
<input type="checkbox"/> dbus.socket	D-Bus System Message Bus Socket	Active (running)	Always	Yes
<input type="checkbox"/> dovecot	Init script for dovecot services	Unknown	Yes	Yes
<input type="checkbox"/> e2scrub_all.service	Online ext4 Metadata Check for All Filesystems	Inactive (dead)	Always	No
<input type="checkbox"/> e2scrub_all.timer	Periodic ext4 Online Metadata Check for All Filesystems	Active (waiting)	Yes	Yes
<input type="checkbox"/> e2scrub_reap.service	Remove Stale Online ext4 Metadata Check Snapshots	Inactive (dead)	Yes	No
<input type="checkbox"/> emergency.service	Emergency Shell	Inactive (dead)	Always	No

Click to expand

In the opening form, we fill in the **Service name**, the **service description**, and the **commands to run on startup**. As we don't need to shut this down, we are going to leave empty the "Commands to run on shutdown" field, check **Yes** for Start at boot time option, and click **create**.

The command to run on startup is: `sudo -u user PHP -d session.save_path=/home/user/tmp /home/user/public_html/tiki-realtime.php`

Replace user with your user

☆ Create Systemd Service

Systemd service details

Service name	tikirealtime.service
Service description	Systemd service to start the WS server
Commands to run on startup	<pre>sudo -u user PHP -d session.save_path=/home/user/tmp /home/user/public_html/tiki-realtime.php</pre>
Commands to run on shutdown	

Start at boot time? Yes No

Create

Return to bootup and shutdown actions

Click to expand

After this, you can check the content of your system service, and it should look something similar to the image below:

☆ Edit Systemd Service

Systemd service details

Service name `tikirealtime.service`

Configuration file `/lib/systemd/system/tikirealtime.service`

Systemd configuration

```
1 [Unit]
2 Description=For Josue
3
4 [Service]
5 ExecStart=sudo -u user php -d session.save_path=/home/user/tmp /home/user/public_html/tiki-realtime.php
6
7 [Install]
8 WantedBy=multi-user.target
9
```

Start at boot time? Yes No

Current status `Running with PID 1135`

Save Start Now Restart Now Stop Now Delete

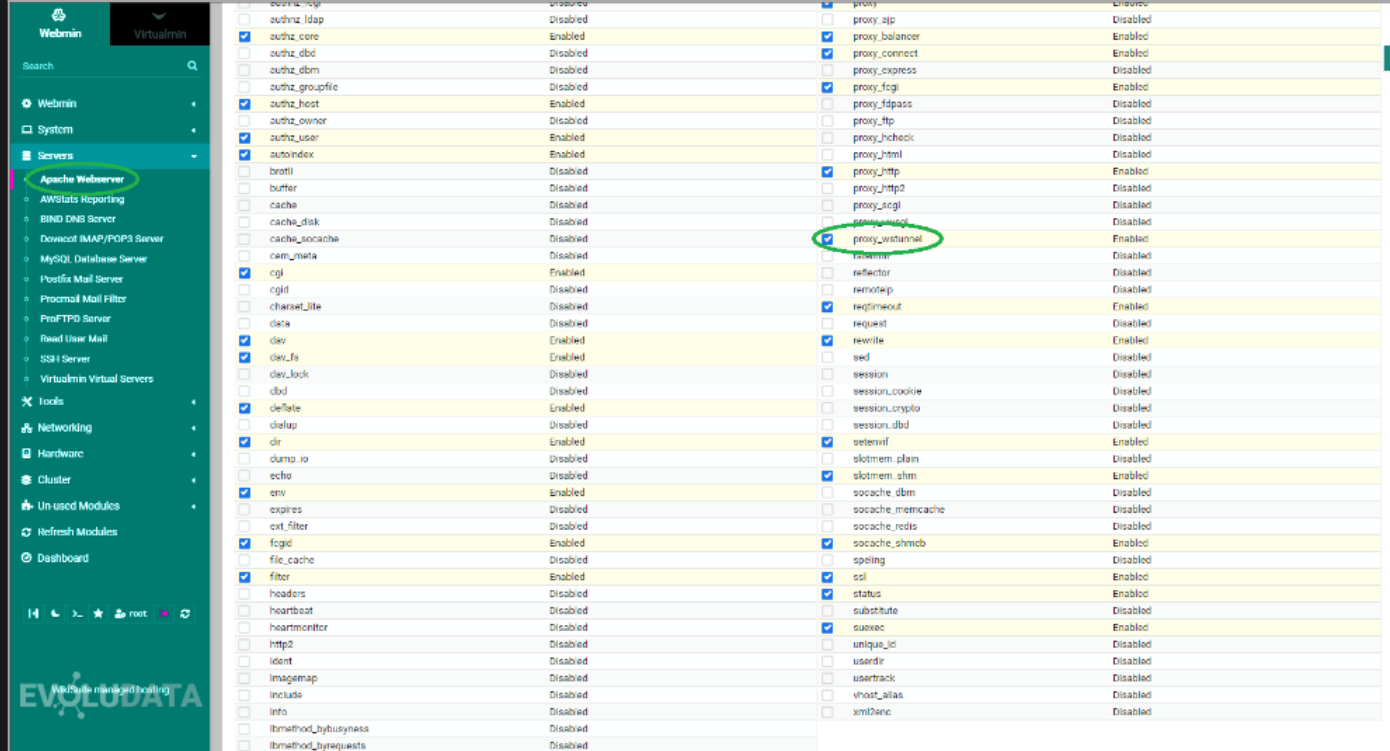
Return to bootup and shutdown actions

Click to expand

3. THE PROXY_WSTUNNEL

The last step here in Virtualmin is to make sure the `proxy_wstunnel` module is enabled.

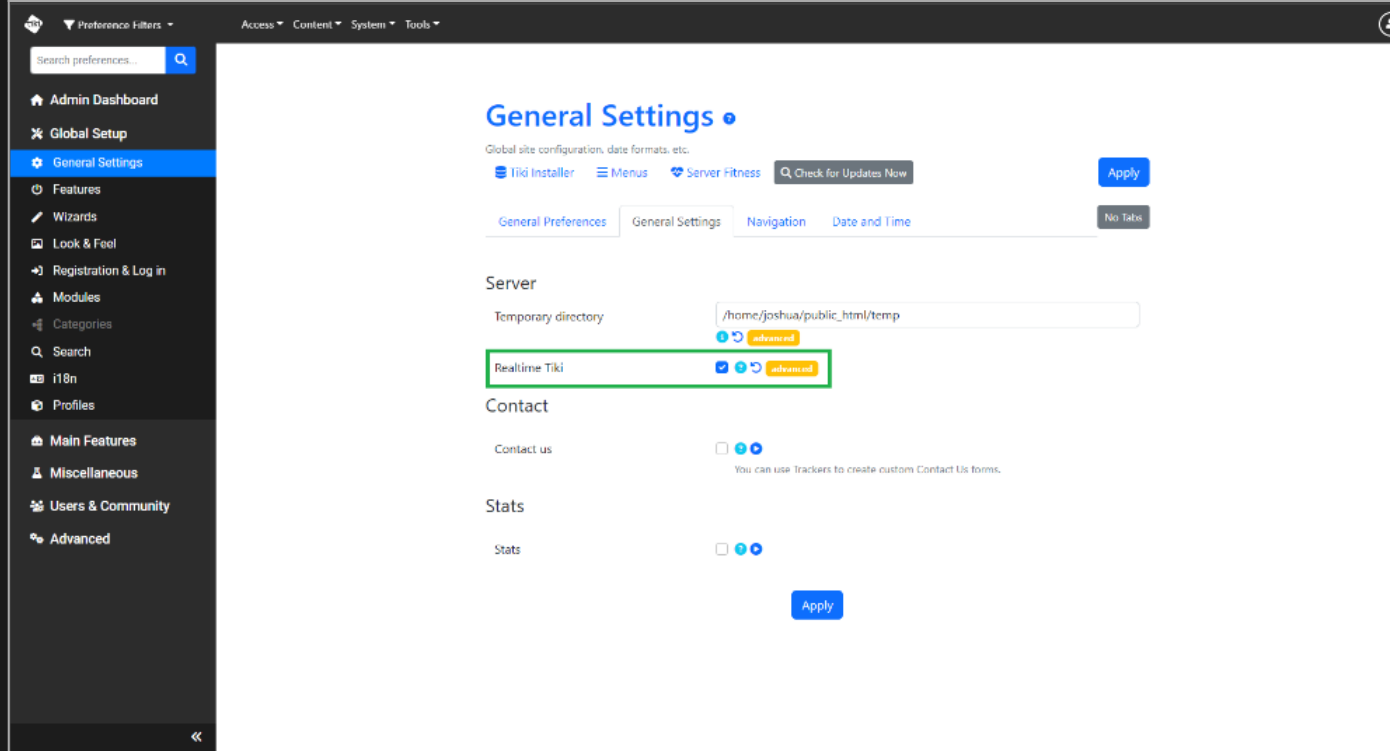
To do this, go to **Webmin** -> **Servers** -> **Apache Webserver** -> **Global configuration** -> **Configure Apache Modules**, and check the **proxy_wstunnel** module checkbox, like in the figure below:



Click to expand

4. REALTIME FEATURE

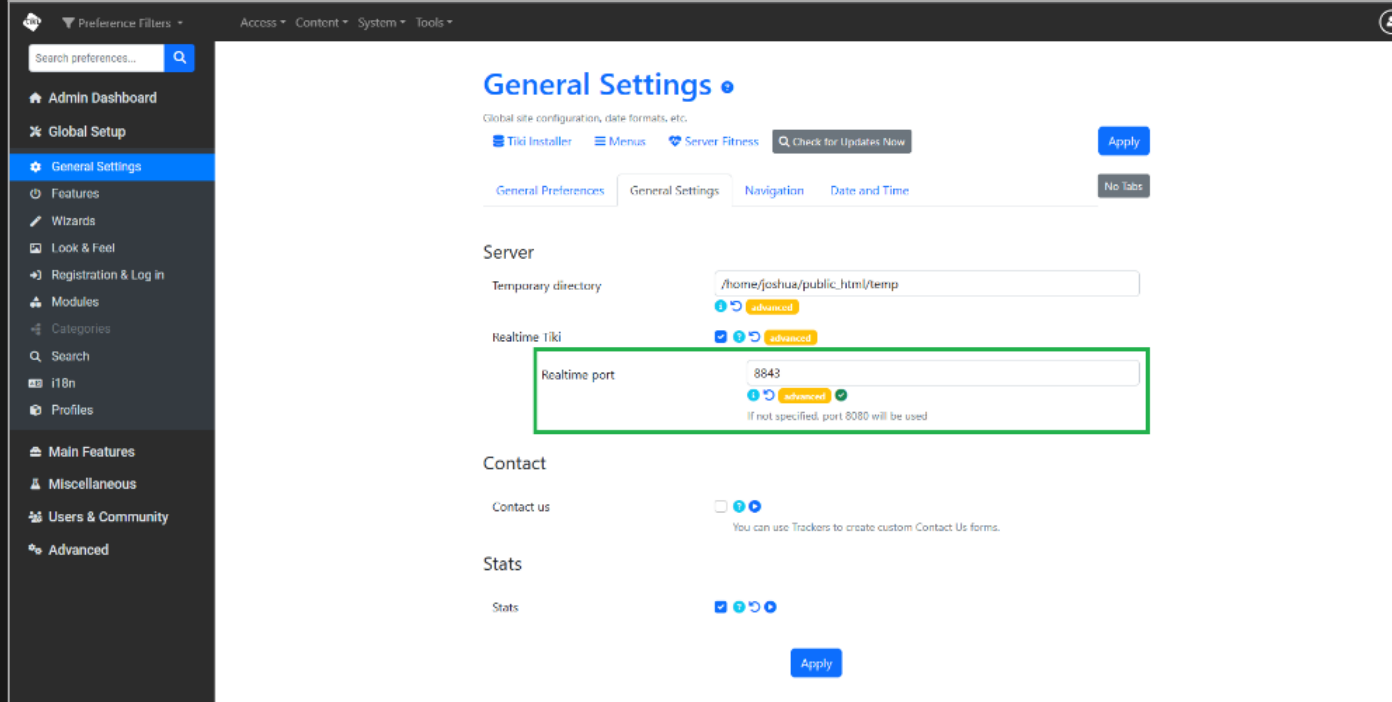
At this step, we need to enable Realtime feature in Tiki. When not enabled, this feature is visible only upon search. Under **Settings** -> **Control Panels**, in the "Search preferences" search box, type a keyword, "realtime", for example.



Click to expand

5. THE REALTIME PORT

Let's note that this not required, but it is an interesting preference and usefull in some cases (default port is not available), because it allows you to set the port on which the server will be listening to. 8080 is the port number by default...



Click to expand

Let's note also that Tiki Realtime can also be started from CLI by running, from the project root, the command ***php tiki-realtime.php***, and, here too, the port can be specified by adding a port parameter to the command: ***php tiki-realtime.php -p 8843*** which means port number will be set to **8843**.

After this is done, Tiki Realtime should be well set up and working. There is a way to check that. So let's see if everything went well...

CHECK THAT REALTIME IS WORKING

This task is as simple as opening our Tiki and running "**Server Check**"

To do this you need to go under **Settings -> Control Panels -> Global Setup -> General Settings** and find "**Server Fitness**", click on it, and on the page that opens, scroll down until you find the "**Tiki Realtime**" section. If Tiki Realtime is working, all the requirements status should be "Good" as shown in the image below:

Realtime Tiki

Requirements	Status	Message
Feature enabled	✔ good	Feature is enabled.
Server listening	✔ good	Server is listening on local system port 8843.
Connectivity	✔ good	Connection to WS server established successfully.
Message exchange	✔ good	Successfully exchanged messages with realtime server.

Click to expand

FOR DEVELOPERS

- Initial commit: https://gitlab.com/tikiwiki/tiki/-/merge_requests/1638
- If you'd like some guidance to deploy this to various Tiki features, please reach out to Marc Laporte

RELATED LINKS

- [Chatbots](#)