Installing Tiki 29 from Git requires more memory, here how to fix on Debian-Ubuntu

Bernard Sfez - 2025-10-09 12:44

If you've tried installing Tiki 29 from GitLab or running setup.sh or npm run build, you might notice something new — your server runs out of memory halfway through. What used to work fine with 2 GB RAM now fails unless you have at least 4 GB or more on your Debian/Ubuntu server.

In this short OpenSource Solutions guide, we'll look at why this happens and how to fix it quickly, either by adding swap memory or upgrading your server's RAM, so you can complete your Tiki 29 VCS installation smoothly.

Why Tiki 29 Needs More Memory

Starting with Tiki 27, the installation and build process uses a more advanced Node.js build system (for compiling CSS/JS assets with Webpack) along with Composer for PHP dependencies. See this article for more information
During setup, these tools run simultaneously and temporarily consume a large amount of memory.

Typical symptoms include:

- Server freezing (most common)
- Process Killed
- npm ERR! code 137
- Out of Memory

The latest Tiki 29, which now requires Node v20.19.5 (LTS), has led to an increase in the memory needed to run the installer or build process.
When the system runs out of available memory, the kernel automatically stops one of the processes — usually Node or PHP — to prevent a crash, effectively aborting the build.
Even though Tiki itself doesn't require much memory at runtime, the installation and build phase now demands significantly more resources.
In short:

- Before Tiki 29: 2 GB RAM was enough.
- Now: You need 4 GB minimum, preferably 6–8 GB for smooth builds.

Upgrade RAM is the best solution, but you can also create or increase swap space as a workaround.

1. Increase Physical Memory

If your hosting provider allows it, the best and most stable fix is to upgrade your VPS or server to 4 GB RAM or more.
This ensures a smooth setup, faster builds, and better overall stability once Tiki is running.

However, this option isn't always as simple as it sounds:

- It costs more — higher memory means a higher hosting plan.
- It's not always smooth — some providers (like AWS) require creating a new instance instead of just resizing, meaning you'll have to migrate your configuration and data.
- It takes time — recreating a server, restoring backups, and testing can interrupt your workflow.

Upgrading is a great long-term choice if budget and time aren't an issue, but if you want a faster, zero-cost workaround, creating or expanding swap memory is the next best solution.

2. Check if You Already Have Swap Memory

The next best solution is to add swap memory and this swap memory allows your system to temporarily use disk space as extra memory instead of physical RAM.

Before creating a swap file, check what's active:

Check if a swap memory exist and its size

```
sudo swapon --show
sudo free -h
```

If Swap shows 0B, or less than 4 GB, you'll need to create or increase it.

3. Create or Increase a 4 GB Swap File (Debian / Ubuntu)

If you don't have any swap file yet:

Create a 4Gb swap memory

```
sudo fallocate -l 4G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
```

If you already have a smaller swap file:

Turn it off:

Turning off the swap memory

```
sudo swapoff /swapfile
```

Delete it:

Delete swap memory

```
sudo rm /swapfile
```

Recreate it with the larger size using the same commands above.
Then verify it’s active:

Check if a swap memory exist and its size

```
sudo swapon --show
sudo free -h
```

You should now see something like:
`Swap: 4.0G`

4. Make the Swap Permanent

To ensure swap remains active after a reboot, add it to /etc/fstab:

Make swap memory permanent

```
sudo echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

Since disk space is inexpensive, there’s no reason not to make this permanent.
You can also fine-tune how often Linux uses swap, it is called the swapiness.

Swap memory swapiness

```
sudo sysctl vm.swappiness=10
sudo echo 'vm.swappiness=10' | sudo tee -a /etc/sysctl.conf
```

This keeps your system responsive while still benefiting from the extra virtual memory.

5. Test and Verify

Reboot your server, then confirm swap is still active.

Check if a swap memory exist and its size

```
swapon --show
free -h
```

Now you can safely rerun your Tiki 29 installation setup or build your files — for example, after modifying templates or adjusting your Tiki site design.
If you’d like help optimizing or personalizing your Tiki environment, our team can handle everything from setup to design integration.

The process should now complete without memory errors.
If it still fails, you likely have a different issue unrelated to memory.

6. Why This Matters — and How We Can Help

The new Tiki 29 build system brings better performance, cleaner assets, and a more modern development workflow — but at the cost of higher memory requirements during installation.
Without enough memory or properly configured swap, you risk failed installs, timeouts, or inconsistent builds.

At Open Source Solutions, we help you:

- Install or upgrade Tiki 29 smoothly on Debian 12 and other Linux servers
- Configure and optimize swap memory permanently
- Tune PHP, Node, and database performance for stability
- Upgrade hosting resources without downtime

If you want your Tiki 29 installation to “just work”, let us handle the setup and optimization.
At OpenSource Solutions, we go beyond installation — we customize Tiki sites to match your business logic, workflows, and brand colors, ensuring your platform looks great and fits your organization perfectly.
Contact us today to ensure your Tiki server runs reliably — with the right memory, proper configuration, and zero frustration.