Engineer

# Linux

the kernel most servers run on

## What it is

Linux is a free, Unix-like operating system kernel: the program that starts first, drives the hardware, shares processor time and memory among every other program, and decides which of them may touch which file or network port. In everyday speech the word stretches to cover the whole operating system assembled around that kernel, and some people write GNU/Linux for the combination, to credit the GNU tools that usually come with it.

Linus Torvalds started it in 1991 as a Unix clone written from scratch, and a loose network of developers working over the internet has built it ever since. It aims at the POSIX and Single UNIX Specification standards, so software written for one Unix-like system usually builds on it with little change. It began on the 32-bit Intel 386 and now runs on a long list of 32-bit and 64-bit architectures. The code is licensed under the GPL, version 2 only, with an explicit note that programs which merely call into the kernel are not covered by it. That note is why proprietary software runs on Linux without a licensing question.

The kernel alone is nothing you can log in to. A working system also needs a C library, a shell, a service manager, a package manager and thousands of programs, and the projects that assemble all of that into something installable are called distributions. Debian, Ubuntu, Red Hat Enterprise Linux and its rebuilds, SUSE, Fedora and Alpine are distributions. Each ships its own build of the kernel, its own defaults and its own promise about how long it will keep fixing security problems.

Its rivals depend on the job. On servers it is weighed against Windows Server and against the BSD family, FreeBSD and OpenBSD, which are complete Unix-like systems developed as one piece rather than assembled from parts. On laptops it meets Windows and macOS.

Use the concept in a decision

Put “What is Linux?” to work in a real decision. We can help relate the concept to the system you are building or buying. Day-to-day care of servers, mail, accounts and updates, so the things everybody relies on stay boring.

**Explore our services** Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Ran 10 PostgreSQL and MS SQL servers on Ubuntu — 10 database servers deployed and kept healthy

**Ran 10 PostgreSQL and MS SQL servers on Ubuntu** Get in touch →

## Why it matters

For a company that runs software on its own servers or in a cloud, the operating system has mostly been chosen before anyone asks. Web servers, databases, container engines and monitoring tools are built for Linux first and tested there hardest. The decision that remains is narrower, and it is worth getting right.

Linux versus Windows Server comes down to the software, not the system. A business built on Active Directory, IIS or older .NET applications has real reasons to stay on Windows. Nearly everything else in a modern web stack, from PostgreSQL and Nginx to Kubernetes, assumes Linux, and even Microsoft SQL Server has a Linux edition.

The cost case is plain. The kernel and community distributions such as Debian and Ubuntu cost nothing to install on as many machines as you like, and there is no licence count to audit; Windows Server is sold under a commercial licence. With Linux the money goes on people and, if you want one, on a support subscription from a vendor such as Canonical, Red Hat or SUSE.

Linux containers tie the knot tighter. Each image packages a Linux user space into a file, and running it means running ordinary processes on a Linux kernel. On a Mac or a Windows laptop, tools running these containers therefore start a small Linux virtual machine first; Apple’s own `container` tool runs each one in a lightweight virtual machine of its own. Choosing Linux containers brings Linux into the stack, whatever the developers’ laptops run.

What actually separates two Linux servers is the distribution and how long it is maintained. The kernel project releases a new mainline version every nine to ten weeks and keeps a handful of longterm branches for years: 6.12, for instance, is projected to receive fixes until December 2028. Distributions build their own

maintenance on top, and a distribution's kernel is supported by the distribution, not by the kernel developers. Pick the distribution for its support period and for the skills the team already has.

One rule of the kernel project makes upgrades less risky than that pace suggests: a change that breaks software which used to work counts as a regression, and regressions get fixed, often by reverting the change. Programs built long ago keep running on new kernels because of it.

The price of all this choice is that "we run Linux" describes very little. Two distributions differ in package manager, firewall tool, network configuration and release rhythm, and a team that knows one has to relearn parts of the other.

## What it looks like in practice

The closest view is a single 464 MB host running a forge and a web server, where the alarm for running out of memory or disk is a dead-man's switch on a timer. Every fifteen minutes, with jitter, a script measures available memory and free disk and pings an outside service only while both clear their thresholds, so silence is the alert. The script had always read the right figure, available memory, while six variable names around it said "free". Because the kernel spends idle memory on cache, free memory on a healthy Linux box sits near zero: against a ten per cent threshold a reader would have seen twelve megabytes free on the 464 MB machine, and a one-character "fix" would have turned a working alarm into one that breaches permanently. The names were corrected. A received signal confirms that the script ran and its available-memory and free-disk thresholds passed; it does not certify general host health or that the site is serving. Monitoring can expose warning signs, although the timing depends on the workload and the conditions observed.

In an earlier role, the founder built an organisation's first observability dashboard, which collected CPU, RAM, HDD, temperature and other metrics from remote Linux servers over SSH, and watched Docker containers the same way. A second version moved to Grafana and Prometheus. It ran on a large TV in the office, with alerts on threshold breaches, and teams detected and resolved incidents 40% faster.

Across a whole estate, access became something written down rather than remembered. The Linux servers were one of the systems that 1,000 RBAC rules governed, alongside developer environments, application instances and the

PostgreSQL and MS SQL databases, and Ansible applied the rules consistently across environments. Developers could reach only their designated servers. The delivered result was a documented role-based rule set with a repeatable way to apply and review it.

## Where it goes wrong

**The dashboard shows memory almost full on a server that is perfectly healthy.** Linux treats unused memory as wasted and fills it with cached file data, which it hands back the moment a program asks. The “free” figure counts only memory doing nothing at all, so it drifts towards zero on any busy machine. The figure that answers “can this server take more work?” is MemAvailable, the kernel’s own estimate of what new programs could use without swapping. Alert on that one:

```
grep -E '^(MemFree|MemAvailable):' /proc/meminfo
```

**A service disappears, and the explanation sits in the kernel log rather than its own.** By default Linux lets programs reserve more memory than the machine has, on the bet that much of it will never be touched. When the bet loses and memory truly runs out, the out-of-memory killer chooses a process, usually a large one, and ends it. The application gets no chance to write anything; the record is in the kernel log, which `journalctl -k` shows. Give each service a memory ceiling of its own, through systemd’s `MemoryMax=` or a container limit, so a runaway program is stopped before the kernel has to choose, and watch available memory instead of waiting for the kill.

**A kernel bug report sent upstream goes unanswered.** The machine runs its distribution’s kernel, which carries the distribution’s own patches, and the kernel developers support only their own releases. `uname -r` shows the version, and anything after the dash, as in `6.8.0-45-generic`, marks a distribution build. Report the problem to the distribution first, or reproduce it on a current kernel from kernel.org before taking it upstream.

**Security fixes stop arriving while the server carries on as before.** Every distribution release and every longterm kernel has an end date, and the machine itself does not announce it. The longterm 5.10 and 5.15 branches, for example,

are projected to reach theirs in December 2026. Keep the end-of-support date in the same inventory as the server's name, and plan the upgrade one release ahead rather than at the deadline.

## Further reading

- About Linux Kernel — the kernel project's own short account of what Linux is and why a newcomer wants a distribution.
- Releases — mainline, stable and longterm kernels, with the projected end of every longterm branch.
- The /proc Filesystem — where MemFree, MemAvailable and the other memory counters are defined.
- Overcommit Accounting — the three overcommit modes and what each one promises a program.
- Reporting regressions — the "no regressions" rule, and how to report a breakage so it counts.
- Linux kernel licensing rules — GPL-2.0 only, and the syscall note that leaves programs on top of it free.



Glossary

Choose for your actual constraints

Bring the deployment path, the operational risks and the people on call. We can scope a change with checks, a recovery route and clear ownership.

**See what it costs** Get in touch →

# Make the terminology practical

Ask how “What is Linux?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

**Explore our services**

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/linux/