**Engineer**

# Downtime

the stretch where a service answers nobody

## What it is

Downtime is time during which a service is out of action for the people who depend on it. Its other names are outage and unavailability; its opposite is uptime, and the share of time a service is up is its availability, usually quoted as a percentage over a month or a year.

The clean case is total: the site does not load, the API refuses connections, the database is down. The common case is messier. A checkout that times out while the catalogue works, a login page that loads but rejects every password, an API answering one request in three: each is downtime for whoever needs that path, even though a health check pointed at the home page would report green. A useful definition counts what users can do, not whether a process is running.

Downtime comes in two kinds. Unplanned downtime is the outage nobody chose: a disk, a bad deploy, an expired certificate, a dependency that went down first. Planned downtime is chosen in advance, usually inside a maintenance window, to upgrade, migrate or move something that cannot change while it serves traffic. Planned downtime is easier to live with, since users can be warned, but it still counts against availability unless the agreement explicitly excludes it.

HTTP has a vocabulary for the moment itself. RFC 9110 defines the 5xx class of status codes for a server that knows it has erred or cannot perform what was asked. Three of them mean different things during an outage. 502 Bad Gateway says a proxy got an invalid answer from the server behind it. 504 Gateway Timeout says the proxy waited too long for one. And 503 Service Unavailable is the one to send on purpose: the specification ties it to a temporary overload or scheduled maintenance, and lets the server say when to come back:

```
HTTP/1.1 503 Service Unavailable
Retry-After: 120
```

That tells browsers, API clients and search engine crawlers the condition is temporary and suggests waiting two minutes, which is a very different message from a broken page served with a success code.

Use the concept in a decision

Put “What is downtime?” to work in a real decision. We can help relate the concept to the system you are building or buying. Monitoring and alerts that speak up while a fault is still small, and the figures that show how often it stays that way.

**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. Built the company's first observability dashboard — the company's first view of its own systems

**Built the company's first observability dashboard** Get in touch →

## Why it matters

Availability targets are written in nines, and the nines translate into time. Over a 365-day year, which is 8,760 hours, each target allows this much downtime:

- 99%: 87.6 hours a year, about three and a half days;
- 99.9%: 8.76 hours a year, or 43.2 minutes in a 30-day month;
- 99.95%: 4.38 hours a year, or 21.6 minutes in a 30-day month;
- 99.99%: 52.56 minutes a year;
- 99.999%: about 5.26 minutes a year.

Each extra nine cuts the allowance by ten, and the cost of earning it rises steeply. At 99.9% a person can be woken, log in, find the problem and fix it inside the monthly budget. At 99.99% the whole year allows less than an hour, which leaves room for perhaps one incident, so recovery has to be automatic. At 99.999% even a routine restart has to happen without users noticing.

Dependencies multiply. A request that needs a load balancer, an application and a database, each available 99.9% of the time and failing independently, is served only when all three are up: $0.999^3$, or about 99.7%, roughly 26 hours of

downtime a year instead of 8.76. A service cannot be more available than the chain of things it needs, unless it removes links from the chain or duplicates them.

Planning starts from the business side. NIST's contingency planning guide for federal systems, SP 800-34, names three numbers worth settling before anything fails. The maximum tolerable downtime (MTD) is the total outage the owner of the system will accept. The recovery time objective (RTO) is how long a system can stay down before the impact becomes unacceptable, and it has to be shorter than the MTD. The recovery point objective (RPO) is how far back in time the data may be restored from the last backup, so it measures tolerable data loss rather than time offline. With those three written down, a backup schedule and a failover design are answers to a question rather than guesses.

## What it looks like in practice

On an IPTV and OTT streaming platform with customers in China, the US and Germany, every hour was someone's prime time, so a quiet hour for maintenance did not exist. Support ran around the clock for roughly a thousand servers plus a good number of client-owned systems. Much of the work was noticing something drifting before it became an outage, because on a live stream an outage is someone's television stopping. The platform stayed continuously available for its global audience.

At a mid-sized company, ten PostgreSQL and SQL Server database servers ran at 99.9% uptime, which by the arithmetic above is an allowance of 8.76 hours a year each. Behind the figure were automated backups with `pg_dump` and SQL Server Agent jobs, with retention policies and offsite storage, monitoring in Prometheus and Grafana, and disaster recovery procedures documented for point-in-time restores and failover.

And on a fleet of thirty Ubuntu servers whose setup had grown by accident, the disaster recovery side was rebuilt with backups and restore procedures that were actually tested. Network configuration became one hardened baseline across every instance. The downtime risk dropped, with recovery paths that had been tested before anyone needed them.

## Where it goes wrong

**The monitor sits inside the outage.** A health check that runs on the same server, or in the same data centre, keeps reporting green while users meet an expired TLS certificate, a broken DNS record or a load balancer pointing at nothing. Probe from outside, from more than one place, along the path a real user takes, and alert on what that probe sees.

**The error page says 200.** A framework that renders a friendly error page with a success status code hides the outage from monitoring, from clients that would retry, and from crawlers, which may index the error page in place of the real one. Errors should carry an error status, and planned maintenance should answer 503 with a Retry-After header.

**The backup was never restored.** A backup proves only that bytes were copied. Until a restore has been run, timed and checked against the data, the recovery time is a guess, and it is often far longer than the RTO on paper. Restore on a schedule, into a separate environment, and write down how long it took. The same applies to failover: a standby that has never taken over is a hope, and rehearsing failover under calm conditions helps expose problems before an incident.

**Releases cause the outages they were meant to prevent.** A deploy that stops the old version before the new one is ready leaves a gap on every release. A schema change that rewrites a large table under an exclusive lock blocks every query against it until it finishes. Roll out new instances behind a readiness check before the old ones stop, split schema changes into an expand step and a later contract step so old and new code both work in between, and on Kubernetes set a PodDisruptionBudget so that a node drain cannot take every replica down at once.

## Further reading

- RFC 9110: HTTP Semantics, section 15.6 — the 5xx status codes, including 503 and its Retry-After hint.
- NIST SP 800-34 Rev. 1: Contingency Planning Guide for Federal Information Systems — where maximum tolerable downtime, RTO and RPO are defined.
- PostgreSQL: High Availability, Load Balancing, and Replication — standby servers and failover, as the database's own manual describes them.

- PostgreSQL: Continuous Archiving and Point-in-Time Recovery — the base backup and WAL archive that a point-in-time restore depends on.
- Kubernetes: Disruptions — voluntary disruptions and the PodDisruptionBudgets that limit them.

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 downtime?” 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/downtime/