Engineer

Disaster recovery

getting service back after a major loss, by a tested plan

What it is

Disaster recovery is the part of contingency planning that covers the worst case: the place a system runs in has been lost or is out of reach for a long time, and the system has to be rebuilt somewhere else from copies. NIST's Contingency Planning Guide for Federal Information Systems, SP 800-34 Revision 1 from 2010, defines the disaster recovery plan as a written plan for recovering one or more information systems at an alternate facility after a major hardware or software failure or the destruction of facilities. Its scope is narrow. The guide reserves it for major disruptions, usually physical ones, that deny access to the primary facility for an extended period, and it covers only the disruptions that require relocation.

That narrowness sets it apart from its neighbours. An information system contingency plan, in the guide's vocabulary, restores a single system wherever it happens to stand, whether after a failed disk or a corrupted database. The disaster recovery plan is tied to a site: it moves operations to the alternate location, and from there each system is brought up by its own contingency plan. A plan for one database and a plan for losing a whole data centre, cloud region or provider account are different documents with different owners, and confusing the two is how a company ends up with neither.

Three measures turn the plan from an intention into something that can be checked. Maximum tolerable downtime (MTD) is the total length of outage the system's owner is willing to accept. The recovery time objective (RTO) is the longest a system resource can stay unavailable before the damage to other systems and to the business becomes unacceptable, and because interrupted work also has to be reprocessed afterwards, the guide says the RTO must normally be shorter than the MTD. The recovery point objective (RPO) looks backwards: it is the point in time before the outage to which data can be

recovered from the most recent backup copy, and it expresses how much lost data the business can live with. Put simply, the RTO is how long you are down and the RPO is how much work you lose.

The alternate site comes in grades. A cold site offers space and infrastructure only, a warm site is partly equipped, and a hot site is fully equipped and staffed. The guide adds mobile sites, which in many cases can be delivered within 24 hours before setup begins, and mirrored sites, the most expensive option and the one it credits with “virtually 100 percent availability”. Cold sites are the cheapest to maintain and may take substantial time to equip. A second cloud region or a second provider plays the same role for most software today, and the trade-off between what stands ready and what it costs to keep it ready is the same.

Use the concept in a decision

Put “What is disaster recovery?” to work in a real decision. We can help relate the concept to the system you are building or buying. Backups that are tested rather than assumed, and a written plan for the morning a server does not come back on.

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. Automated 100 critical data backups using Barman — 100 critical backups automated across the databases

Automated 100 critical data backups using Barman Get in touch →

Why it matters

A backup is a copy, and a copy on its own runs nothing. Getting from the copy to a working system also takes the software that reads it, the keys that decrypt it, servers and a network to run it on, the configuration that ties them together, and someone who knows the order to do it in. Any of those that existed only in the lost site was lost with it. Disaster recovery is the discipline of making sure that all of them survive the same event, and that the path from copy to service has been walked before anyone needs it.

Recoverable data is only as recent as the last copy that restores, whatever RPO the backup schedule is meant to meet. A nightly job that has failed silently for a week leaves a week’s data at risk, whatever the policy document says. NIST’s guide makes the same point about tape, recommending that backups be tested regularly to confirm that the data is stored correctly and that the files can be retrieved without errors or loss. A green status light on a backup job proves that something was written. Only a restore proves it can be read back.

The plan also decays. People leave, servers are replaced, providers redesign their consoles, and the passwords in the runbook expire. Of the seven steps NIST lays out for contingency planning, the last two are testing, training and exercises, and plan maintenance; the guide wants the plan to be a living document, reviewed at least once a year and whenever the system or the resources used for recovery change significantly. A plan that has never been rehearsed tends to fail on details nobody thought to write down: the order the services must start in, a DNS record with a day-long time to live, the one credential that was stored on the machine that is gone.

For software teams, the disaster is often an account rather than a building. A deleted cloud project, a compromised administrator login, a suspended provider account or a destructive command run against production removes access to the system as thoroughly as a flood does. Standby servers are no protection against the last of these, since a standby applies a mistaken `DROP TABLE` within moments, exactly as it applies every other change.

What it looks like in practice

For a PostgreSQL database, a recovery point far shorter than the gap between two backups comes from continuous archiving of the write-ahead log (WAL) on top of a base backup. The server copies each finished WAL segment somewhere safe, and a restore replays those segments onto the base backup up to any chosen moment since the base backup was taken:

```
# postgresql.conf on the protected server
wal_level = replica
archive_mode = on
archive_command = 'test ! -f /mnt/archive/%f && cp %p /mnt/archive/%f'
```

To restore, a base backup is unpacked into a fresh data directory, an empty file named `recovery.signal` is placed in it, and the recovery settings say where the archive is and where to stop:

```
restore_command = 'cp /mnt/archive/%f %p'
recovery_target_time = '2026-03-10 17:14:00+01'
recovery_target_action = 'pause'
```

The PostgreSQL documentation tells the story these settings exist for: someone drops a critical table at 5:15 on a Tuesday evening, notices on Wednesday at noon, and restores to 5:14. Pausing is the default once the target is reached, so that queries can confirm it is the right moment before the server is promoted to accept connections; if it is not, the server is shut down, given a later target and started again. The chosen recovery time must be later than the completed base backup, and the documentation advises setting up and testing the archiving before the first base backup is taken. When recovery completes, the server removes `recovery.signal` so that it does not enter recovery again by accident.

Engineer ApS offers backup and disaster recovery as a service, described as backups that are tested rather than assumed and a written plan for the morning a server does not come back. Two portfolio pages show what that means. In backups and disaster recovery as code, from 2025, backup retention was set in Bicep beside the database, so point-in-time restore is part of the template instead of a setting someone once clicked in a portal, and the environment around the database was defined as code too, because a database restored into an environment rebuilt from memory is hardly recovered. The recovery position was then audited and written down with its gaps named: geo-redundancy was switched off, and the recovery time objective was proposed rather than measured, since no drill had been run.

The company's backup and restore-test automation was designed to extend recovery beyond a provider snapshot. An Ansible role installs restic scripts and systemd units for a database dump where configured, encrypted deduplicated snapshots to storage at another provider over SFTP, retention pruning and integrity checks. A separate monthly workflow restores a known file to a scratch location and compares its bytes. Backup and restore have separate success signals, the passphrase comes from a protected file, and timers depend on a configured repository. Those controls are implemented; a live recovery is a separate deliverable. The implementation record dated 2026-10-27 says the timers were disabled without a repository and records no completed live restore, so it supports the automation work rather than a recovery-time commitment or a claim that the company can already lose the host safely.

Where it goes wrong

The backup has never been restored. A job that reports success every night says nothing about whether its output can be read, decrypted and loaded, and the first real restore is a poor moment to find out. Schedule restores the way you

schedule backups: an automated check that pulls known data out and compares it, and a full restore into a clean environment often enough that the time it takes is a measured number.

The copy shares the original's fate. A snapshot in the same cloud account, a backup disk in the same rack or an archive bucket behind the same administrator login goes down with whatever takes the primary. NIST's guide asks for offsite storage and alternate sites to be chosen for how unlikely they are to be hit by the same disaster. Keep at least one copy with another provider or in another account with separate credentials, and keep the decryption keys somewhere that does not depend on the systems they protect.

Only the data was planned for. Restoring a database is a small part of bringing a service back if the servers, firewall rules, DNS, certificates and secrets around it have to be recreated from memory. Define the environment in version-controlled infrastructure as code, and make rebuilding it part of the drill, not only loading the data.

The objectives are guesses. An RTO written into a document without a drill behind it is a proposal, and an RPO that assumes every backup succeeded is optimistic. Measure both in an exercise. NIST separates tabletop exercises, which are discussion only and deploy no equipment, from functional exercises that simulate operations, up to full-scale ones that cover every element of the plan; the numbers come from the second kind.

A standby is counted as a backup. A standby server or a geo-replicated database protects against losing hardware, but it copies deletions, corruption and files encrypted by ransomware as faithfully as good writes. Keep point-in-time copies that a mistake on the primary cannot reach, and keep them for longer than it usually takes to notice a mistake.

The plan describes last year's system. A runbook that lists retired servers, people who have left and passwords that no longer work slows a recovery down instead of guiding it. Review the plan after every significant change and after every drill, record what the drill got wrong, and review the parts that change most often, such as contact lists, more often still, as NIST also recommends.

Further reading

- SP 800-34 Rev. 1, Contingency Planning Guide for Federal Information Systems — the full guide: how it defines and scopes the DRP, the seven planning steps, MTD, RTO and RPO, alternate sites and exercise types, with plan templates.
- Continuous Archiving and Point-in-Time Recovery (PITR) — PostgreSQL 18's chapter on WAL archiving, base backups and restoring to a chosen moment.
- Write Ahead Log — the configuration reference for `archive_command`, `restore_command` and the recovery target settings.

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 disaster recovery?" 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/disaster-recovery/