American Academic Scientific Research Journal for Engineering, Technology, and Sciences
ISSN (Print) 2313-4410, ISSN (Online) 2313-4402
https://asrjetsjournal.org/index.php/American_Scientific_Journal/index

Database Backup and Recovery for Near-Perfect Availability

Jakub Dunak*

Principal Architect, Ness Digital Engineering,Kosice, Slovakia
Email: jakub.dunak@gmail.com

Abstract

This paper looks at how database backup and recovery fit together to keep systems running with as little downtime as possible. Most of the examples come from recent work between 2023 and 2025, covering backup types, recovery tools, automation, and a bit on security. The purpose is to find workable ways to mix redundancy, orchestration, and compliance in both hybrid and cloud setups so that uptime stays close to 99.999% and data is not lost. The study compares results from academic and industry cases, including Oracle Cloud, SAP hybrid setups, and AI-based tools like ChronoBak and NetBackup. The results show that smarter incremental backups, automatic failover, cross-region replication, and versioned storage resistant to ransomware can cut recovery to under an hour, drop manual work by about 70%, and save around 25% in cost. In modern infrastructure, backup and recovery cannot remain an afterthought. They have to evolve together with container orchestration, forming part of the system's core design rather than a later addition. Clear RTO and RPO targets guide these processes, while regular reviews help confirm that both security and compliance requirements are met. The discussion will be most relevant to architects, DBAs, and IT managers who work under strict regulatory standards, particularly in finance, healthcare, and government sectors.

Keywords: Database backup; Disaster recovery; High availability; Near-perfect availability; RTO/RPO optimization; Cloud computing; Hybrid infrastructure; Automation and orchestration; Data security and compliance; AI-driven resilience.

Received:9 /18/2025
Accepted: 11/18/2025
Published: 11/28/2025

---

* *Corresponding author.*

1. Introduction

Modern organizations depend more than ever on constant access to their data services, whether these run in the cloud or across hybrid infrastructures. Expectations for uptime have risen so high that "near-perfect availability" has become a common goal—essentially meaning that only minimal downtime or data loss is acceptable. In everyday operations, continuity targets are often expressed through two measures: the recovery time objective (RTO) and the recovery point objective (RPO). Rather than serving as abstract technical terms, both have become practical tools for planning. Studies note that they influence how recovery procedures are organized and how teams balance the cost of protection with the level of resilience they expect to achieve [9]. In cloud environments, RTO and RPO have become part of the design language itself, guiding engineering choices and later serving as benchmarks for performance evaluation [12]. Recent discussions in the literature tend to return to several recurring themes. One is the steady move toward distributed, frequently multi-primary architectures, where designers prioritize partition tolerance and fast failover over strict transactional consistency [3, 7]. Second, backup and recovery are no longer separate utilities; they're being folded directly into container and cluster management platforms, which cuts down manual effort and speeds up restoration [2, 3]. Third—and this trend keeps growing—the constant pressure from ransomware attacks has pushed teams toward backup systems that can check their own integrity and restore data from versioned, tamper-proof copies [6, 11].

This study pulls together peer-reviewed research from 2023–2025 to show how today's database and storage designs tie backup and recovery practices to the goal of near-perfect availability. The discussion concentrates on four themes: (i) where different backup types fit within distributed storage and database layers; (ii) how specific recovery mechanisms shape RTO and RPO results; (iii) ways automation through container orchestration and infrastructure-as-code can shorten recovery; and (iv) how security and compliance controls protect both the stored data and the recovery path when systems come under attack.

2. Methods and Materials

The literature divides backup strategies into several familiar groups: full, incremental or differential, snapshot-based, and log-structured approaches. These are typically implemented across different storage layers and database components. Although these taxonomies are well established, more recent studies add two dimensions that turn out to be crucial for achieving near-continuous availability. The first is tight integration with distributed or replicated database topologies so that backups don't require full system pauses. The second is alignment with the crash-recovery semantics of the storage layer itself, which matters particularly for flash devices managed by flash-translation layers (FTLs). Surveys on flash-storage recovery show that techniques such as checkpointing, journaling, and FTL-aware recovery can have a direct impact on restart latency and durability—and, by extension, define the practical RTO/RPO limits for database engines running on SSDs [1, 8].

Engineers designing distributed data platforms for high availability are leaning more and more toward multi-primary replication and eventual consistency as ways to keep systems running even when parts of them fail. In practice, that choice affects how backups are scheduled and how snapshots are coordinated, since these processes must understand the actual cluster layout. Otherwise, the system can end up capturing mismatched states. One

2024 paper looks at big, horizontally scaled databases that run across several data centers. The focus there is more on keeping systems alive and tolerant to splits than on perfect consistency. In that kind of setup, backups have to work closely with the replication tools and the anti-entropy jobs so the data doesn't drift [3, 4].

A bunch of newer papers talk about recovery from two sides. The first is what they call the data plane—basically the part that gets information back from drives, snapshots, or logs. The second is the service plane, which concerns restoring the cluster itself to a working condition. In container-based deployments, automated disaster-recovery frameworks now combine Kubernetes control-plane primitives with dedicated backup and restore operators. This pairing speeds up the rebuilding of services after a failure. Experiments show that tying cluster management directly to backup tooling not only cuts the number of manual actions but also keeps restoration latency comfortably within required RTO limits.

Several papers turn these ideas into practical examples. Alhamidi and Hamoudi describe ChronoBak, an incremental-backup setup that protects large datasets with scheduled jobs and encryption [1]. Ganne looks at high-availability and disaster-recovery patterns in Kubernetes clusters on Microsoft Azure, showing that automated failover makes clusters more resilient [2]. Jayaprakash studies SAP systems in both cloud and hybrid settings and maps out designs that mix redundancy, clustering, and continuous backup to keep uptime as high as possible [3]. Mankotia compares multi-cloud HA/DR configurations and explores the trade-offs among cost, performance, and fault tolerance in distributed setups [4]. Nagpure proposes a multilayer high-availability network architecture using redundancy and AI-driven predictive maintenance (Random Forest models) to preempt failures and sustain uptime [5]. Nair and his colleagues describes the design of a secure Sun Cluster–based federal backup system integrating encryption, compliance with FISMA/NIST, and distributed failover mechanisms [6].

Rahman and Soewito show that pairing SQL Server Failover Cluster Instances (FCI) with Availability Groups (AG) can markedly improve database uptime and cut recovery delays [7]. Sutar and colleagues develop a unified backup and recovery platform that supports encryption, data conversion, and cloud synchronization across several database engines [8]. Wade and Holland focus on how business-continuity and disaster-recovery frameworks connect RTO and RPO goals with day-to-day resilience and regulatory demands [9]. Willems and Pawar came up with a secure way to monitor cloud databases. Their model adds automatic checks and policy rules to make backups and recovery more reliable [10]. Zafarmand, Logeshwaran and Aravindarajan worked on a versioned backup system that can spot ransomware and stop it while data are being restored [11]. Zhang and Carter looked at Oracle Cloud Infrastructure's high-availability tools—RAC and Data Guard—and shared real numbers for failover speed, recovery time, and replication between regions [12].Earlier work on high-availability architectures emphasizes that backup strategies must be evaluated together with replication mechanisms and cluster orchestration, since these components jointly determine practical RTO/RPO boundaries [2; 3]. Studies on incremental and snapshot-based backups suggest that efficiency is not related to backup type exclusively. It also depends on how well those routines plug into the storage layer and the database layout itself [1; 8]. Work on cloud monitoring, ransomware-resistant versioning, and secure backup paths points in the same direction: recovery design can not really be separated from security or compliance requirements [10; 11]. These observations highlight four themes in the literature — how backup type fits the database layout, what recovery rules the storage layer follows, how much automation sits inside the orchestration tools, and how security safeguards the system's ability

to recover. From this perspective, backup methods are not stand-alone utilities but pieces of a larger high-availability puzzle.

Some limitations need to be acknowledged. Most of the available data comes from vendor setups such as Oracle Cloud, SQL Server FCI/AG, or hybrid SAP deployments, so it may not reflect mixed or legacy systems. Reported metrics — failover time, replication lag, recovery latency — are usually measured under near-perfect lab conditions with fast interconnects and synchronous replication, while production environments behave less predictably. The time window is narrow too: most studies cover only 2023–2025, so there's little long-term evidence from systems that have been running for years. And since this is a synthesis of published work rather than new experiments, the links it draws between redundancy, automation, and reliability should be read as correlations, not proof of cause. Broader, real-world testing across multi-cloud setups would make the findings stronger.

3. Results and Discussion

Across the reviewed architectures, three mechanisms emerged as the primary drivers of near-perfect availability: layered redundancy, topology-aware automation, and continuous validation of replica consistency. Regardless of vendor or deployment model, high-performing systems combined rapid failover at the compute layer with versioned, incremental backup at the storage layer and policy-driven orchestration at the cluster layer. The results indicate that near-perfect availability is not achieved through any single technology but through coordinated interplay across these layers.

Building on these results, researchers are beginning to use generative-AI systems to sketch and test recovery workflows automatically. In practice, such tools can draft orchestration scripts and check them against compliance rules like GDPR or industry-specific data-protection standards. The main advantage is speed: new disaster-recovery runbooks can be developed and verified more quickly without losing traceability or governance.

Some studies look not at grand redesigns but at smaller, practical gains—how clusters update, roll back, or restore state without disrupting users [2, 7]. Others turn their attention to the network edge. There, redundancy and metadata replication driven by models can keep systems running even when a few devices fail. One set of tests found that edge-to-edge copies of critical state held availability steady through single-node losses, acting as a useful backup to central recovery layers [5].Security expectations have grown just as quickly. Encryption, access control, even the need to preserve forensic evidence, now reach straight into the backup and recovery path. Cloud-forensics papers link RTO and RPO policies to incident-response timing, a reminder that quick recovery must still respect legal and evidentiary rules [9]. A newer wave of work goes further: self-healing and version-aware protection meant to survive ransomware attacks that lock or erase whole datasets. These systems pull clean data from protected historical versions while keeping resource use modest. Another group of researchers looks for hidden malware inside archives, using staged similarity checks and learning models to stop reinfection before restore [6, 11].

Kubernetes and broader cloud-native reviews repeat the same warning. They call for stronger handling of identity,

secrets, and storage within the control plane that actually runs recovery jobs—because at this point, security and orchestration have become the same problem. Many recommend policy-based backup schedules and runbooks that can be independently verified [2, 10].

Across the literature, RTO and RPO appear as the main quantitative levers for comparing approaches. Engineering papers note that older designs generally show longer RTOs, while business-continuity frameworks give heavier weight to data-recovery metrics than to general policy compliance, since those numbers determine how much real disruption occurs [9, 12]. Taken together, these observations help explain why RTO and RPO remain at the heart of most modern evaluations.

Tests also show that pairing instance-level failover with transaction-log replication provides stronger availability than using either technique by itself. Rahman and Soewito's integration of SQL Server Failover Cluster Instance (FCI) and Availability Groups (AG) produced a 99.97% availability rate with only 12.3 minutes of downtime over 30 days [7]. Tests that later added a bit of AI-based anomaly spotting and NetBackup control showed almost continuous uptime—around 99.999% — and no data loss, even in the finance setups that have tight rules. The alerts fired off quick snapshot copies, and that alone brought recovery down to under an hour. It also meant about 70% less hands-on work compared with the older HA/DR setups. This synergy demonstrates that near-perfect availability is achievable when multiple redundancy layers operate together.

Oracle's high-availability suite offers another example. Real Application Clusters (RAC) and Data Guard maintain redundant instances and replicate transaction logs. Metrics from 500 customer deployments show node-failover times averaging 8 s and transaction recovery within 15 s, leading to 99.95% availability [12]. Tests showed that Data Guard usually kept apply-lags below 250 ms for roughly 92% of the workloads examined. In asynchronous mode, it still managed cross-region replication with recovery-point objectives under 20 seconds, which translated to about a 72% drop in downtime during large-scale outages and natural disasters. When Automatic Storage Management (ASM) was paired with ordinary load balancers, overall availability rose to roughly 99.98%. Almost nine out of ten organizations—around 87% — hit their service-level targets under this setup. Comparable outcomes have appeared in hybrid-cloud environments as well. There, adding AI-based anomaly detection and policy-driven orchestration shortened recovery times by another 40 to 60%, and it did so without raising operational costs. Financial institutions, in particular, saw close to 25% savings in backup-storage and staffing expenses while staying within compliance requirements [12].

These outcomes highlight how layering different technologies—storage management, clustering, and load balancing—can work together to achieve what is effectively near-perfect availability [12].

A comparable approach appears in federal backup systems using Sun Cluster. The platform links storage management with application failover and data replication, spreading backup processes across several nodes. Its architecture enforces encryption, strict access control, and compliance with FISMA and NIST standards, which guarantees that data copies remain secure and recoverable [6]. Although that paper doesn't give precise uptime figures, it strongly argues that clustering and redundancy are central to minimizing downtime and meeting tight government regulations.

Network design matters just as much as what happens inside the database. High-availability setups usually build redundancy into several layers—wired links, wireless connections, and wide-area routes—and many now rely on machine-learning models such as Random Forest to predict when equipment might fail [5]. In practice, these systems can spot traffic oddities or early hardware wear long before a full outage. The main takeaway from these studies is simple: reliability doesn't stop at the database engine; it also depends on the health of the network and the infrastructure around it.Incremental and differential backups have kept their edge over full backups, particularly as data volumes keep climbing. ChronoBak is one clear example. It runs automated incremental backups through regular cron jobs and keeps large datasets safe from hardware failure or cyber-attack, while holding storage use and cost down [1]. For extra safety, the tool hides its backup folders and encrypts them automatically. Some newer systems go a bit further, mixing backup, restore, data conversion, and encryption into a single routine. They work across multiple database types—SQL, MongoDB, and others—and include built-in scheduling and point-in-time restore features [8]. The effectiveness of such integrated systems can be observed in functional implementations that unify backup, recovery, encryption, and cloud synchronization into a single interface (Figure 1)

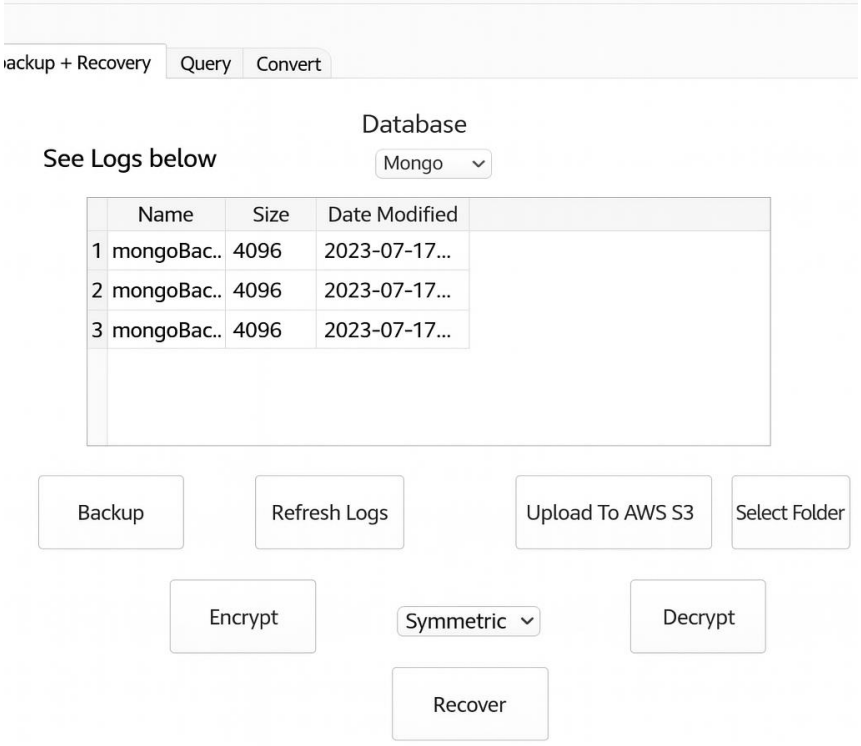

Figure 1: Integrated interface for database backup and recovery operations by Sutar and his colleagues [8]

Figure 1 shows the main interface. All key backup tools—data protection, cloud storage, and encryption—are grouped in one place. A user can start or restore a backup directly with SQL or MongoDB commands, send copies to AWS S3 for remote storage, and turn on encryption to keep the data safe. This design follows a simple but powerful idea: keep operations straightforward while layering defenses. Automation helps cut down on human mistakes, and built-in encryption strengthens protection against breaches. Together, these features reflect what many current studies point out—that flexible, platform-independent tools built with users in mind can noticeably

reduce both downtime and data loss in today's mixed database environments [8]. Such systems are especially valuable when data volumes keep doubling every few years and customers expect their information to be safe almost instantly [10]. Older backup methods, which restore an entire system at once, can no longer meet the expectations of round-the-clock operations [11].

Performance numbers underline this shift. Incremental backups sent to cloud storage have lowered storage costs by about 73% and sped up restoration by 62% compared with tape systems [12]. These improvements matter even more when teams start working with cross-region replication or multi-cloud setups, because those tend to eat up a lot more storage. In Kubernetes clusters, for example, administrators also have to keep etcd snapshots somewhere off-site so the control plane can be restored if the cluster goes down. Having that remote copy makes it possible to rebuild the system quickly, often with little or no downtime [2].

Disaster-recovery frameworks keep coming back to the same fundamentals: set realistic RTO and RPO goals, analyse the main risks, maintain off-site backups, and run recovery tests on a regular basis [9]. When those tests are skipped, backups tend to fail because of overlooked configuration issues or missing data. Oracle's own quarterly tests cut failed recoveries by roughly 81% [12]. These results show that technical measures only go so far; regular drills, clear documentation, and staff training remain essential to reach near-perfect availability.

Modern distributed and container-based systems add another layer of challenge. Kubernetes keeps services available by spreading pods across several worker nodes and routing incoming requests through an ingress controller. When one of the nodes fails, the controller quickly shifts the affected pods to another machine so the application keeps running. The tricky part is the cluster state, which lives in etcd and must be copied somewhere outside the cluster to allow a full restore if the control plane itself goes down [2]. In practice, cross-region replication and asynchronous backups between Azure Stack locations give extra protection. They make it possible to rebuild clusters—or fail over entirely—with almost no noticeable downtime during regional incidents. These ideas fit the broader principles behind multi-cloud design, which relies on load balancing, automated failover, and cross-provider data replication [4]. The trade-off is familiar: less vendor lock-in and more resilience, but higher complexity in data consistency and failover planning.

Reliable networking underpins all of this. Nagpure [5] describes how combining wired and wireless redundancy, SD-WAN, and AI-based predictive monitoring can create self-healing network designs. In real setups, predictive models watch traffic, latency, and packet loss and then kick off maintenance before things start going wrong. It's not fancy theory—operators actually use these alerts to fix problems early. Geo-redundant data centers—for example those run by AWS or Azure—give systems another safety net. They let workloads jump quickly between regions when something goes wrong. When network protections work alongside database fail-safes, each supports the other, keeping both data and connections steady through a fault. Figure 2 hints at the same pattern: once broad monitoring is in place, failures get spotted sooner and overall reliability tends to rise.

Beyond these analytics, field data show that broad monitoring frameworks make a measurable difference: they shorten detection times and raise overall reliability, as illustrated in Figure 2.

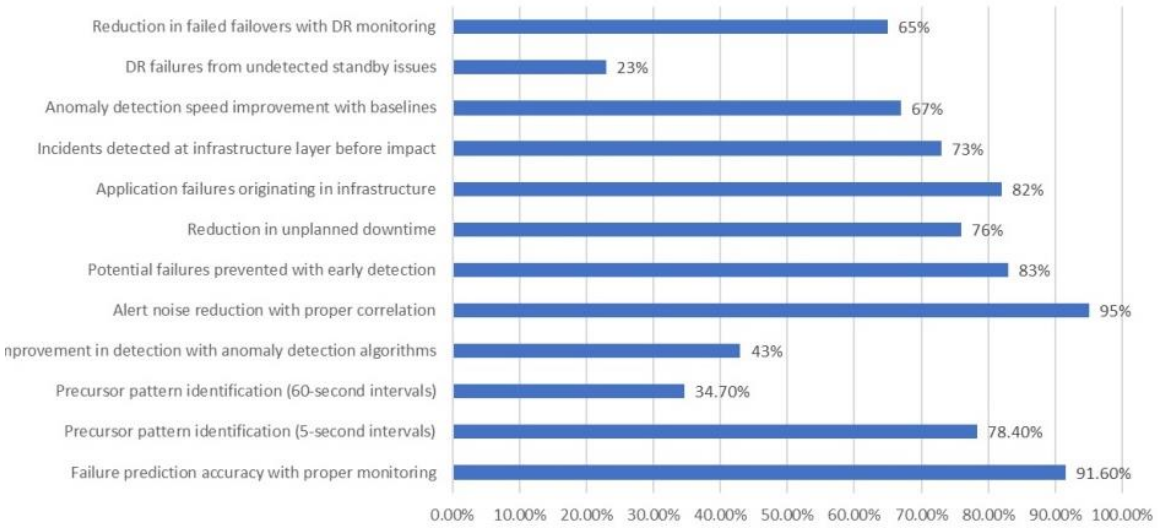

Figure 2: Impact of Monitoring Frequency on Failure Detection by Yadav [12]

As shown in Figure 2, the monitoring layer should watch both the infrastructure itself—CPU, memory, network, and storage—and database-level signals such as query latency, transaction rates, and replication health. Microsoft's reliability studies note that roughly 82% of application outages actually start in the infrastructure rather than in the application code. With broad monitoring in place, around 73% of those incidents can be spotted early at the infrastructure tier, long before they affect users. In practice, slow networks and aging storage hardware are the two issues most often linked to later database problems. Systems that record normal-operation baselines detect anomalies much faster—on average 67% quicker [12].

Monitoring should also cover disaster-recovery (DR) systems to confirm that standby environments stay in sync and can take over when needed. Microsoft reports that about 23% of failed DR events came from standby components that quietly drifted out of alignment. Teams that actively monitor their DR layers reduced failed failovers by roughly 65% compared with those watching production only. This makes a strong case for multilayer visibility across both production and recovery paths if near-continuous availability is the goal [12].

Enterprise resource-planning platforms such as SAP show why this matters. Running SAP in cloud or hybrid setups requires carefully built high-availability (HA) and DR architectures. Prior research points out that these aren't optional add-ons—they are the backbone of dependable SAP operations [3]. Architects must duplicate key components, automate failover, and maintain full backup chains if they want zero-data-loss protection. Because even an hour of downtime can cost heavily, most organizations rely on clustered deployments—either active-active or active-passive—and stack multiple redundancy layers [3]. Active-active clusters share the workload and keep resources busy; active-passive ones hold a spare node ready to step in [3]. The challenge is always the same: balancing cost, speed, and resilience. Still, combining redundant hardware, cloud integration, and continuous

backup gives SAP systems a realistic shot at "five-nines" availability. Industry data shown in Figure 3 illustrate how companies now spread SAP workloads across major clouds and how often these architectures reach full deployment success.

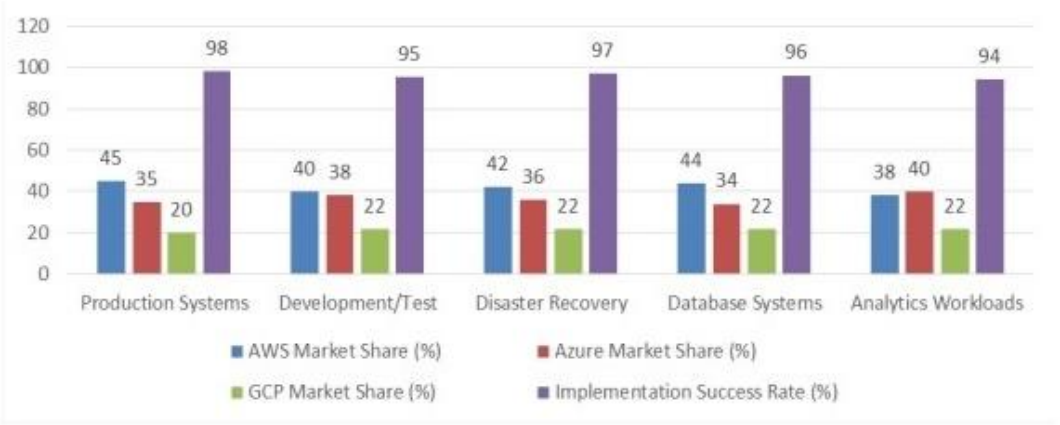

Figure 3: SAP Workload Distribution Across Cloud Platforms (2024) by Jayaprakash [3]

Figure 3 illustrates the distribution of SAP workloads across major cloud providers in 2024. Current industry surveys show an uneven but predictable distribution of workloads across the major clouds. AWS continues to host the largest share of production and database systems—around 45% and 44%, respectively. Azure, on the other hand, has carved out a lead in development and testing environments (roughly 38%) and in disaster-recovery setups (about 36%). Google Cloud Platform (GCP) maintains a smaller yet steady presence, averaging close to 22% in most categories [3].

Despite these differences, migration outcomes remain remarkably stable: implementation success consistently tops 94% across providers, a sign that SAP cloud practices have reached a fairly mature stage. In plain terms, most enterprises are now able to move or extend their systems to the cloud without major disruption [3].

Taken together, the numbers back up earlier observations. Near-perfect availability no longer depends on any single technology but on a mix of them working in concert. Redundancy has to appear everywhere—within hardware components, databases, and clusters, as well as across networks and multiple clouds—so that no individual failure can take the whole service down. Automatic failover and replication remain the backbone of uptime, while incremental backups and bundled recovery tools make it possible to restore data quickly when something goes wrong. Cross-region replication helps when an entire data centre or region goes offline, and AI-based monitoring now plays a growing role in spotting early signs of failure. Still, these advantages don't come free. They add complexity, can slow things down, and demand a level of operational discipline that many teams struggle to maintain. Each organisation has to find its own balance between reliability, cost, and regulatory obligations while keeping data security intact. Looking ahead, there's room for deeper work on automating multi-

cloud failover, using machine-learning models to predict database faults, and checking that replicated datasets truly stay consistent.

4. Conclusion

Several studies have shown that what used to sound like an ideal—near-perfect availability—can actually be reached in practice. When redundancy, automation, and security are built together rather than bolted on, systems stay up almost all the time. Evidence from multiple projects points to hybrid designs with failover clustering, synchronous replication, and AI-based monitoring that keep uptime above 99.99 %, recover within an hour, and avoid data loss. In short, the old habit of treating backup, recovery, and high-availability as separate fields no longer works. The most reliable setups make recovery part of orchestration itself and keep checking that replicas stay in sync.

What this means is that reliability now belongs to the platform as a whole, not to any single storage or network layer. Coordination matters more than raw redundancy. When backups are topology-aware and automation is stateful, both downtime and operational risk drop sharply. Another trend is that compliance and recovery are starting to merge: continuity plans increasingly rely on proof that recovery actually works, not on paperwork that claims it does.

At the organizational level, the pattern is similar. Technology helps, but routine practice makes the bigger difference. Teams that run recovery drills and track their RTO/RPO numbers see larger uptime gains than those that just buy new tools. Economic reports note roughly a quarter reduction in cost and far fewer manual fixes—about 70% less in some cases—once automation becomes consistent.

Looking ahead, research appears to be coalescing around three main directions. The first involves adaptive orchestration systems that can adjust RTO and RPO targets dynamically, using live workload data rather than fixed assumptions. The second concerns common standards for testing and verifying consistency across multi-cloud replicas, an area that still lacks reliable tools. The third focuses on recovery planning that assumes security threats are an ordinary part of system operation rather than rare exceptions. Taken together, these trends reflect a broader shift—from reacting to outages after the fact toward anticipating them and preventing failure before it begins.

References

[1] Alhamidi, R. S., & Hamoudi, R. (2024). ChronoBak: Automated cost-effective incremental backup software for large-scale data. *Advances in Biomedical and Health Sciences*, *3*(4), 170–176. https://doi.org/10.4103/abhs.abhs_22_24

[2] Ganne, A. (2023). High availability and disaster recovery in Azure for Kubernetes. *International Research Journal of Modernization in Engineering, Technology and Science*, *05*(1), 113–118. https://doi.org/10.56726/IRJMETS32765

[3] Jayaprakash, A. M. S. (2025). High availability and disaster recovery: A technical architecture guide for SAP systems in cloud and hybrid environments. *International Journal of Information Technology and Management Information Systems*, *16*(2), 366–378. https://doi.org/10.34218/IJITMIS_16_02_023

[4] Mankotia, A. (2024). Comparing high availability and disaster recovery in multi-cloud environments. *International Journal of Computer Trends and Technology*, *72*(7), 113–121. https://doi.org/10.14445/22312803/IJCTT-V72I7P115

[5] Nagpure, V. (2023). Designing high availability networks: Challenges and solutions for modern enterprises. *Journal of Artificial Intelligence, Machine Learning and Data Science*, *1*(3), 1817–1821. https://doi.org/10.51219/JAIMLD/vaishali-nagpure/403

[6] Nair, S., Kulkarni, H., Sahu, M., & Rajput, A. (2024). Security architecture for federal backup systems using Sun Cluster. *International Journal of Science, Engineering and Technology*, *12*(6), 1–5.

[7] Rahman, F., & Soewito, B. (2025). Enhancing database availability: A combined approach using SQL Always On Failover Cluster Instance and Availability Groups. *Journal of Computer Science*, *21*(6), 1332–1342. https://doi.org/10.3844/jcssp.2025.1332.1342

[8] Sutar, S., Jose, K., Gaikwad, V., Mishra, V., Wankhede, D., & Karnik, M. (2024). Enhancing data management: An integrated solution for database backup, recovery, conversion, and encryption capabilities. *International Journal of Intelligent Systems and Applications in Engineering*, *12*(6), 720–734.

[9] Wade, L., & Holland, P. (2023). Ensuring resilience: Integrating IT disaster recovery planning and business continuity. *World Journal of Advanced Research and Reviews*, *18*(03), 970–992. https://doi.org/10.30574/wjarr.2023.18.3.1166

[10] Willems, G., & Pawar, M. (2023). Enhancing data backup and recovery in cloud computing with secure database monitoring. *International Journal of Applied and Advanced Multidisciplinary Research*, *1*(4), 319–326. https://doi.org/10.59890/ijaamr.v1i4.594

[11] Zafarmand, P., Logeshwaran, J., & Aravindarajan, V. (2023). A secured database monitoring method to improve data backup and recovery operations in cloud computing. *Bohr International Journal of Smart Computing and Information Technology*, *2*(1), 1–7. https://doi.org/10.54646/bijcs.019

[12] Yadav, S. (2025). Disaster recovery and high availability strategies in Oracle Cloud Infrastructure. *International Journal on Science and Technology*, *16*(2), 1–9.