Post Incident Review

Backups Unavailable After Data Loss Incident

Joe R.

Table of Contents

Case Study: Backups Unavailable After Data Loss Incident

Executive Summary

This summary is based on a real-world incident in which a production database was accidentally deleted by an employee attempting to resolve an ongoing outage. Although the company had implemented multiple backup strategies, none of them functioned properly when restoration was attempted. In this document, I will review the company's public statements on their backup procedures and examine the likely points of failure in their operational response.

Incident Review and Expert Commentary

Multiple Backups In Place

> **Quote:** "This brings us to the recovery procedures. Normally in an event like this, one should be able to restore a database in relatively little time using a recent backup, though some form of data loss can not always be prevented. For GitLab.com we have the following procedures in place:
>
> - Every 24 hours a backup is generated using pg_dump, this backup is uploaded to Amazon S3. Old backups are automatically removed after some time.
> - Every 24 hours we generate an LVM snapshot of the disk storing the production database data. This snapshot is then loaded into the staging environment, allowing us to more safely test changes without impacting our production environment. Direct access to the staging database is restricted, similar to our production database.
> - For various servers (e.g. the NFS servers storing Git data) we use Azure disk snapshots. These snapshots are taken once per 24 hours.
> - Replication between PostgreSQL hosts, primarily used for failover purposes and not for disaster recovery."

Commentary:

There is a significant amount of technical language in the company's statement describing various backup methods. Despite the presence of multiple backup strategies, none were effective in facilitating recovery. The only valid backup solution for the database was the pg_dump method. This approach integrates directly with the database, capturing all transactions in a consistent state and allowing for safe restoration.

Snapshots offer a fast and convenient way to capture the entire server that supports the database; however, they are not endorsed by the database vendor as a reliable recovery method. To illustrate, consider a database processing millions of transactions per minute—capturing all of them at a single point in time is beyond the capability of a basic snapshot.

Replication was correctly identified as a mechanism for failover only, not as a disaster recovery solution.

Case Study: Backups Unavailable After Data Loss Incident

Supported Backups Were Failing, Likely Due to an Upgrade

> **Quote:** "When we went to look for the pg_dump backups we found out they were not there. The S3 bucket was empty, and there was no recent backup to be found anywhere. Upon closer inspection we found out that the backup procedure was using pg_dump 9.2, while our database is running PostgreSQL 9.6 (for Postgres, 9.x releases are considered major). A difference in major versions results in pg_dump producing an error, terminating the backup procedure."

Commentary:
What's notably omitted from the outage disclosure is whether the pg_dump backups had ever functioned correctly. It is likely they worked at some point but broke during the upgrade from PostgreSQL 9.2 to 9.6. Typically, database and backup teams operate independently, and in all likelihood, the database team proceeded with the upgrade without coordinating with the backup team. This points to a potential breakdown in change management and a lack of cross-functional collaboration within the organization.

No One Knew Backups were Failing

> **Quote:** "While notifications are enabled for any cronjobs that error, these notifications are sent by email. For GitLab.com we use DMARC. Unfortunately DMARC was not enabled for the cronjob emails, resulting in them being rejected by the receiver. This means we were never aware of the backups failing, until it was too late."

Commentary:
Email alerts are a common method for tracking backup successes or failures, but they are far from the only option. Even if emails were being filtered as spam due to DMARC issues, both failure and success notifications would have been affected. The absence of *any* alerts should have raised concerns.

Furthermore, a backup administrator typically spends most of their time in a backup console, which would have clearly indicated the failed status of this backup. This suggests a lack of organizational awareness and accountability with respect to backup monitoring and responsibility.

Backups Were not Tested

> **Quote:** "**Why was the backup procedure not tested on a regular basis?** - Because there was no ownership, as a result nobody was responsible for testing this procedure."

Commentary:
This confirms our earlier suspicion that the organization lacked a formal backup program aligned with modern compliance standards. Both NIST and ISO frameworks require regular testing of backup procedures. The absence of assigned ownership for this critical function highlights a significant deficiency in professional organizational control regarding business continuity and disaster recovery.

Preventive Controls and Organizational Responsibility

This incident illustrates critical deficiencies in how backup responsibilities were managed. While multiple backup methods were technically present, none were functional when recovery was needed. These gaps indicate a failure to meet widely accepted practices found in both NIST and ISO compliance frameworks for data protection and business continuity.

1. Lack of Regular Backup Testing

The organization did not routinely test its backup processes. When the primary backup method failed, it had already been inoperable for some time. Formal programs under NIST and ISO require periodic testing of backup data to ensure it can be successfully restored. The absence of testing shows a breakdown in operational discipline and risk preparedness.

2. Missing Backup Monitoring and Alerting Oversight

Failures in the pg_dump process were not detected in time. The organization relied exclusively on email alerts, which were silently failing due to misconfigured DMARC settings. Even without email, backup administrators should have visibility into job status through management consoles or monitoring tools. Not seeing any alerts at all should have triggered investigation, yet no follow-up occurred.

3. Poor Coordination During System Changes

The backup process was broken by a PostgreSQL version upgrade that introduced incompatibility between the backup software and the database. This happened without coordination between the database and backup teams. A mature change management process would include impact analysis for critical systems and communication between responsible teams to prevent this type of failure.

4. Misuse of Backup Methods

LVM and cloud-based disk snapshots were in use, but these tools are not considered reliable recovery mechanisms for transactional databases. They do not provide application-consistent or point-in-time guarantees. These tools are valuable for staging or testing, but should not be relied upon as substitutes for validated database backups. Industry guidance makes clear that backup strategies must align with the data types and workloads being protected.

5. Absence of Ownership and Accountability

Perhaps the most concerning failure is the lack of designated ownership over the backup process. No team or individual was responsible for verifying backup success or initiating regular recovery tests. NIST and ISO standards both emphasize clear accountability for data protection functions. Without ownership, even the best tools and procedures will fail when they are most needed.

Appendix A: Company Disclosure

Partial Excerpt: Full Report Available Here.

Broken recovery procedures

This brings us to the recovery procedures. Normally in an event like this, one should be able to restore a database in relatively little time using a recent backup, though some form of data loss can not always be prevented. For GitLab.com we have the following procedures in place:

1. Every 24 hours a backup is generated using `pg_dump`, this backup is uploaded to Amazon S3. Old backups are automatically removed after some time.
2. Every 24 hours we generate an LVM snapshot of the disk storing the production database data. This snapshot is then loaded into the staging environment, allowing us to more safely test changes without impacting our production environment. Direct access to the staging database is restricted, similar to our production database.
3. For various servers (e.g. the NFS servers storing Git data) we use Azure disk snapshots. These snapshots are taken once per 24 hours.
4. Replication between PostgreSQL hosts, primarily used for failover purposes and not for disaster recovery.

At this point the replication process was broken and data had already been wiped from both the primary and secondary, meaning we could not restore from either host.

Database backups using pg_dump

When we went to look for the `pg_dump` backups we found out they were not there. The S3 bucket was empty, and there was no recent backup to be found anywhere. Upon closer inspection we found out that the backup procedure was using `pg_dump` 9.2, while our database is running PostgreSQL 9.6 (for Postgres, 9.x releases are considered major). A difference in major versions results in `pg_dump` producing an error, terminating the backup procedure.

Appendix A: Company Disclosure

Partial Excerpt: Full Report Available Here..

The difference is the result of how our Omnibus package works. We currently support both PostgreSQL 9.2 and 9.6, allowing users to upgrade (either manually or using commands provided by the package). To determine the correct version to use the Omnibus package looks at the PostgreSQL version of the database cluster (as determined by `$PGDIR/PG_VERSION`, with `$PGDIR` being the path to the data directory). When PostgreSQL 9.6 is detected Omnibus ensures all binaries use PostgreSQL 9.6, otherwise it defaults to PostgreSQL 9.2.

The `pg_dump` procedure was executed on a regular application server, not the database server. As a result there is no PostgreSQL data directory present on these servers, thus Omnibus defaults to PostgreSQL 9.2. This in turn resulted in `pg_dump` terminating with an error.

While notifications are enabled for any cronjobs that error, these notifications are sent by email. For GitLab.com we use DMARC. Unfortunately DMARC was not enabled for the cronjob emails, resulting in them being rejected by the receiver. This means we were never aware of the backups failing, until it was too late.

Why was the backup procedure not tested on a regular basis? - Because there was no ownership, as a result nobody was responsible for testing this procedure.

Appendix B: Relevant Security Control Categories

These controls are considered standard for most large organizations.

The following controls, derived from NIST 800-53 (Rev. 5) and ISO/IEC 27001:2022, were likely deficient or missing in the incident response described. Their absence contributed materially to the organization's inability to restore systems after a database loss.

1. Backup Monitoring and Alerting

- **NIST Control**: **AU-5(1)** – *Response to Audit Processing Failures*
- **ISO Control**: **Clause 5.7** – *Monitoring, measurement, analysis, and evaluation*
- **Expected Practice**: Backup failures should generate alerts through reliable and monitored channels, not just email. Alerts must be reviewed and escalated when critical jobs fail.
- **Observed Gap**: Cronjob alerts were configured via email only, and misconfigured DMARC policies caused silent alert suppression. No backup dashboard or secondary monitoring path existed.

2. Regular Testing of Backup and Restoration

- **NIST Control**: **CP-9(1)** – *Testing of Information System Backup*
- **ISO Control**: **A.5.30** – *ICT readiness for business continuity*
- **Expected Practice**: Backup systems must be tested regularly to confirm they can be restored in practice, not just assumed to be functional.
- **Observed Gap**: pg_dump backups had been silently failing due to a version mismatch for an extended period, and no restore testing uncovered the issue.

3. Assignment of Ownership and Responsibility

- **NIST Control**: **PM-13** – *Information Security Roles and Responsibilities*
- **ISO Control**: **Clause 5.3** – *Organizational roles, responsibilities, and authorities*
- **Expected Practice**: All critical procedures, including backup and recovery, must have assigned ownership with documented responsibilities.
- **Observed Gap**: No one was assigned to verify or maintain the backup process. The absence of ownership led to prolonged undetected failure.

4. Change Management and System Compatibility

- **NIST Control**: **CM-4** – *Impact Analysis for Changes*
- **ISO Control**: **A.8.32** – *Change management*
- **Expected Practice**: All system changes should include an assessment of their impact on related systems, including backup processes.
- **Observed Gap**: The upgrade from PostgreSQL 9.2 to 9.6 broke pg_dump compatibility. Backup scripts continued to run, but failed silently due to a lack of change impact assessment.

5. Coordination Between Operational Units

- **NIST Control**: **CA-1** – *Coordination Among Organizational Entities*
- **ISO Control**: **Clause 6.1.3** – *Information security risk treatment*
- **Expected Practice**: Teams responsible for critical interdependent systems must coordinate during upgrades, maintenance, and risk treatments.
- **Observed Gap**: Database and backup teams operated in silos. The database team likely performed the upgrade without informing or coordinating with those responsible for backups.

6. Valid Backup Strategy and Data Recovery Suitability

- **NIST Control**: **CP-10** – *System Recovery and Reconstitution*
- **ISO Control**: **A.5.29** – *Information deletion* and **A.8.13** – *Information backup*
- **Expected Practice**: Backup solutions must be appropriate for the systems they protect. Snapshots are not a valid substitute for logical, application-aware database backups.
- **Observed Gap**: Snapshots were incorrectly relied upon as potential recovery options despite vendor documentation not supporting them for transactional integrity.