Preventing Full Disk Issues

Running out of disk space in a database environment can lead to failed writes, service downtime, and even data corruption. PostgreSQL systems require available space not only for storing data but also for managing temporary files, WAL logs, indexes, and routine background tasks. On Elestio, while infrastructure is managed, you are still responsible for monitoring growth and preventing overuse. This guide outlines how to monitor disk usage, configure alerts, automate cleanup, and follow best practices to avoid full disk conditions in PostgreSQL.

Monitoring Disk Usage

Proactively monitoring disk usage helps you detect unusual growth in time to act. Whether you're accessing your database directly via the terminal or through a Docker Compose environment, several built-in tools can provide usage stats and trends. Combining filesystem-level monitoring with PostgreSQL-specific checks gives a complete view of space utilization

To check the overall disk usage of the system from a terminal or container:

```
df -h
```

This command shows available space for each mounted volume. Focus on the mount point where your PostgreSQL data directory is stored, usually `/var/lib/postgresql`.

For detailed PostgreSQL-specific usage, connect to your database and run:

```
SELECT pg_size_pretty(pg_database_size(current_database()));
```

This shows the total size used by the active database. You can also analyze individual tables and indexes using:

```
SELECT relname AS object, pg_size_pretty(pg_total_relation_size(relid)) AS size
FROM pg_catalog.pg_statio_user_tables
ORDER BY pg_total_relation_size(relid) DESC
LIMIT 10;
```

This query highlights the largest tables by size, helping you identify which parts of your schema consume the most space.

Configuring Alerts and Cleanup

Even with monitoring in place, automatic alerts and cleanup scripts ensure you act before hitting disk limits. You can set up external monitoring agents or run container-level scripts to track disk usage and notify you.

If you're using Docker Compose, you can monitor container-level storage stats using:

```
docker system df
```

This command provides an overview of Docker volumes, images, and container usage. To monitor and clean unused volumes and logs manually:

```
docker volume ls
docker volume rm <volume-name>
```

Make sure you're not deleting active database volumes. Always verify that backups exist and are up-to-date before running cleanup commands.

To configure PostgreSQL-specific cleanup, enable auto-vacuum and monitor its effectiveness. PostgreSQL removes dead tuples and reclaims space using this process. Check the vacuum activity with:

```
SELECT relname, n_dead_tup, last_vacuum, last_autovacuum
FROM pg_stat_user_tables
ORDER BY n_dead_tup DESC;
```

If dead tuples accumulate, increase autovacuum frequency or run a manual vacuum:

```
VACUUM ANALYZE;
```

Autovacuum settings can also be tuned in postgresql.conf to trigger more aggressively based on table activity.

Best Practices for Disk Space Management

- Beyond immediate cleanup, long-term strategies help keep disk usage under control. These include data retention policies, partitioning, compression, and regular maintenance. It's also important to have a growth plan based on usage trends.

- Avoid storing large binary objects like images or PDFs directly in the database. Use object storage for large files and reference them by URL. If historical data is no longer needed for queries, archive it into a separate cold-storage database or export to files.
- Partition large tables by time or ID ranges to manage growth and make pruning easier. Use tools like `pg_partman` native PostgreSQL table partitioning to automatically offload older data
- Regularly rotate and clean up PostgreSQL logs and WAL files. If using archive mode, ensure archived WALs are uploaded and removed from disk after successful backup.
- To keep your setup safe, also monitor backup file sizes and locations. Backups stored on the same volume as the database may consume critical space. If possible, push backups to remote object storage or another disk volume.

Revision #1
Created 2025-04-09 09:43:24 UTC
Updated 2025-04-09 10:51:17 UTC