# Preventing Full Disk

Running out of disk space in a KeyDB environment can result in failed writes, background save errors, and degraded availability. KeyDB, like Redis, uses disk storage for persistence (RDB and AOF), temporary files, and logs especially when persistence is enabled. On managed hosting platforms like Elestio, while infrastructure maintenance is handled, it is up to the user to monitor disk space, configure retention settings, and perform regular cleanups. This guide walks through how to monitor disk usage, configure alerts, remove unnecessary data, and apply best practices for avoiding full disk issues in a KeyDB setup under Docker Compose.

# Monitoring Disk Usage

Disk usage monitoring helps identify abnormal growth patterns and prevents outages due to insufficient storage. In Docker Compose environments, both host-level and container-level monitoring are essential.

## Inspect the host system storage

Run the following command on the host to check overall disk usage and determine which mount point is filling up:

```
df -h
```

This displays disk usage statistics across volumes. Locate the mount point corresponding to your KeyDB data volume—typically something like /var/lib/docker/volumes/keydb_data/_data.

## Check disk usage from inside the container

To get insight into the container’s internal disk usage, first enter the container shell:

```
docker-compose exec keydb sh
```

Once inside the container, assess the size of the data directory with:

```
du -sh /data
```

This reveals the total size used by KeyDB data files, such as appendonly.aof, dump.rdb, and temporary files. You can also list file-level details with:

```
ls -lh /data
```

This helps identify which files are occupying the most space.

# Configuring Alerts and Cleaning Up Storage

Monitoring disk usage is not enough; you must also set up alerts and take action to reclaim space. On the host system, analyze Docker resource usage with:

```
docker system df
```

This provides insights into how much space is consumed by images, volumes, and containers.

## Identify unused Docker volumes

To list all volumes on the host, run:

```
docker volume ls
```

If you find a volume that is unused and safe to delete, remove it with:

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

Make sure you do not delete the volume mapped to your KeyDB data directory unless it is backed up and verified to be unused.

## Trigger AOF file compaction

When using AOF persistence, the append-only file may grow large over time. You can reduce its size by triggering a background rewrite:

```
docker-compose exec keydb keydb-cli BGREWRITEAOF
```

This creates a compacted version of the AOF file with the same dataset.

## Clean up old snapshots

RDB snapshots accumulate over time if not managed. They are stored in the /data directory inside the container. To list them, run:

```
docker-compose exec keydb ls -lh /data
```

Remove old .rdb files with:

```
docker-compose exec keydb rm /data/dump-<timestamp>.rdb
```

Ensure that any snapshot you remove is not needed for recovery.

# Managing and Optimizing Temporary Files

KeyDB creates temporary files during fork operations, such as when saving snapshots or rewriting AOF files. These are typically stored in /tmp inside the container.

## Monitor temporary file usage

You can inspect the size of the temporary directory with:

```
docker-compose exec keydb du -sh /tmp
```

If this directory becomes full, forked operations like BGSAVE or BGREWRITEAOF may fail. To mitigate this, you can change the temporary directory path in keydb.conf to use a volume-backed location like /data:

```
dir /data
```

Restart the container after making this configuration change.

# Best Practices for Disk Space Management

Effective disk space management in KeyDB depends on adopting a forward-looking configuration and consistent housekeeping.

Avoid storing large binary blobs directly in KeyDB. Instead, keep files like PDFs, images, and other large media in external object storage and only store metadata or keys in KeyDB.

If persistence is not required, disable it entirely to reduce disk usage. This is useful for cache-only workloads:

```
appendonly no
save ""
```

To avoid uncontrolled AOF file growth, configure rewrite thresholds in keydb.conf:

```
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
```

Set up log rotation if your container logs to files such as /var/log/keydb/keydb-server.log. This can be managed using logrotate on the host system, or via Docker logging options in docker-compose.yml:

```
logging:
  driver: "json-file"
  options:
    max-size: "10m"
    max-file: "3"
```

Always use TTLs for cache or session keys to avoid indefinite storage growth. For example:

```
SET session:<id> "data" EX 3600
```

Track memory and persistence statistics with:

```
docker-compose exec keydb keydb-cli INFO memory
docker-compose exec keydb keydb-cli INFO persistence
```

Backup files stored in /data should be offloaded to a remote location. Use Elestio’s built-in backup options or mount a dedicated remote volume using your docker-compose.yml to ensure backups do not consume host disk space indefinitely.

---

Revision #1
Created 2025-06-26 06:56:03 UTC
Updated 2025-06-26 07:19:12 UTC