



An introduction to Garage

Alex Auvolat, Deuxfleurs

<https://garagehq.deuxfleurs.fr/>

Matrix channel: [#garage:deuxfleurs.fr](https://matrix.to/#/#garage:deuxfleurs.fr)

A non-profit initiative



Part of a degrowth initiative

Garage has been created at Deuxfleurs, where we experiment running Internet services without datacenter on commodity and refurbished hardware.



Developed by a community

Some recent contributors: Arthur C, Charles H, dongdigua, Etienne L, Jonah A, Julien K, Lapineige, MagicRR, Milas B, Niklas M, RockWolf, Schwitzd, trinity-1686a, Xavier S, babykart, Baptiste J, eddster2309, James O'C, Joker9944, Maximilien R, Renjaya RZ, Yureka...



Owned by nobody

AGPL + no Contributor License Agreement = Garage ownership spreads among dozens of contributors.

Our initial objective at Deuxfleurs

**Promote self-hosting and small-scale hosting
as an alternative to large cloud providers**

Why is it hard?

Resilience

we want good uptime/availability with low supervision

Our very low-tech infrastructure

- Commodity hardware (e.g. old desktop PCs)
(can die at any time)
- Regular Internet (e.g. FTTB, FTTH) and power grid connections
(can be unavailable randomly)
- **Geographical redundancy** (multi-site replication)

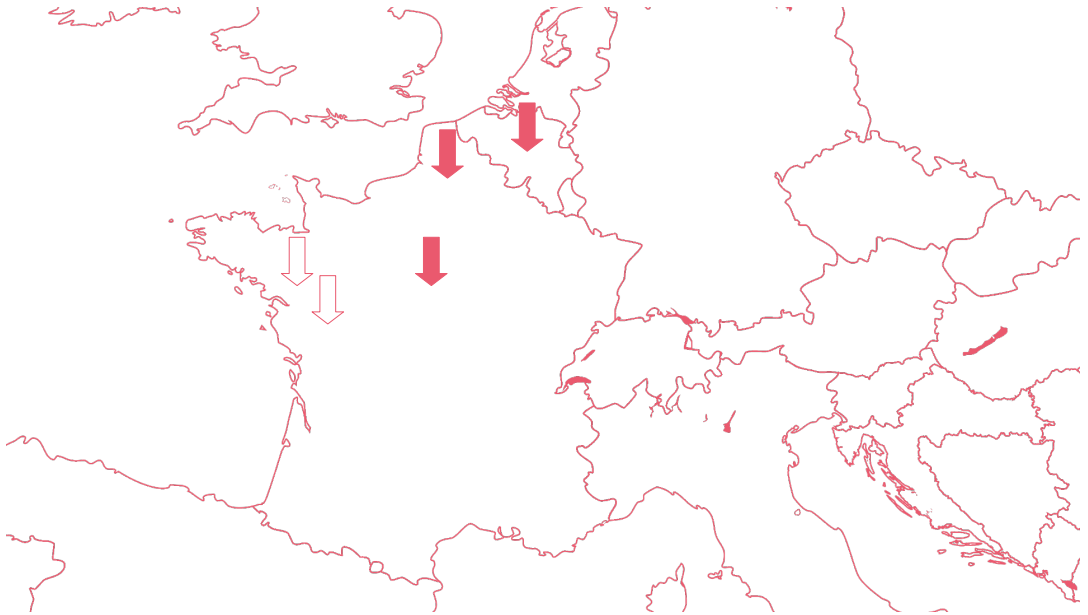
Our very low-tech infrastructure



Our very low-tech infrastructure



Our very low-tech infrastructure



Object storage: a crucial component

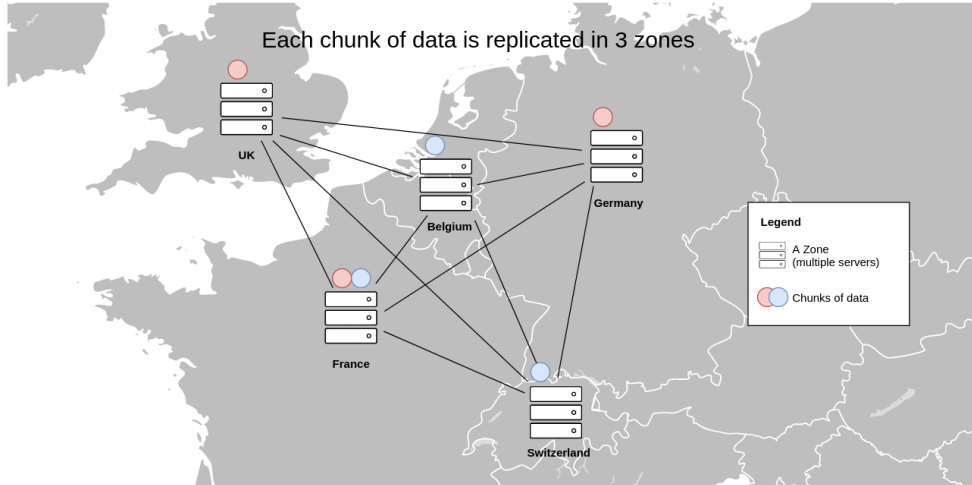


S3: a de-facto standard, many compatible applications

MinIO: not suited for geo-distributed deployments, becoming closed source

Garage is a self-hosted drop-in replacement for the Amazon S3 object store

Principle 1: geo-distributed data model



Garage stores replicas on different zones when possible

Zone-aware cluster configuration

```
[root@celeri:/home/lx]# docker exec -ti e338 /garage status
==== HEALTHY NODES ====
ID                Hostname  Address                               Tags                               Zone  Capacity
5fcb3b6e39db3dcb  concomb  [2001:470:ca43::31]:3901            [concomb,neptune,france,alex]    neptu  500.0 GB
942dd71ea95f4904  df-ymf   [2a02:a03f:6510:5102:6e4b:90ff:fe3a:6174]:3901 [df-ymf,bespin,belgium,max]     bespin  500.0 GB
fdfaf7832d8359e0  df-ymk   [2a02:a03f:6510:5102:6e4b:90ff:fe3b:e939]:3901 [df-ymk,bespin,belgium,max]     bespin  500.0 GB
0a03ab7c082ad929  ananas   [2a01:e0a:e4:2dd0::42]:3901        [ananas,scorpio,france,adrien]  scorpio 2.0 TB
a717e5b618267806  courgette [2001:470:ca43::32]:3901          [courgette,neptune,france,alex]  neptune 500.0 GB
2032d0a37f249c4a  abricot  [2a01:e0a:e4:2dd0::41]:3901        [abricot,scopio,france,adrien]  scorpio 2.0 TB
8cf284e7df17d0fd  celeri   [2001:470:ca43::33]:3901          [celeri,neptune,france,alex]    neptune 2.0 TB
17ee03c6b81d9235  df-ykl   [2a02:a03f:6510:5102:6e4b:90ff:fe3b:e86c]:3901 [df-ykl,bespin,belgium,max]     bespin  500.0 GB
```

Trust model: full trust between zones

Principle 2: based on CRDTs

Internally, Garage uses only CRDTs (conflict-free replicated data types)

Why not Raft, Paxos, ...? Issues of consensus algorithms:

- **Software complexity**
- **Performance issues:**
 - The leader is a **bottleneck** for all requests
 - **Sensitive to higher latency** between nodes
 - **Takes time to reconverge** when disrupted (e.g. node going down)

The data model of object storage

Object storage is basically a **key-value store**:

Key: file path + name	Value: file data + metadata
index.html	Content-Type: text/html; charset=utf-8 Content-Length: 24929 <binary blob>
img/logo.svg	Content-Type: text/svg+xml Content-Length: 13429 <binary blob>
download/index.html	Content-Type: text/html; charset=utf-8 Content-Length: 26563 <binary blob>

Consistency model:

- Not ACID (not required by S3 spec) / not linearizable
- **Read-after-write consistency**
(stronger than eventual consistency)

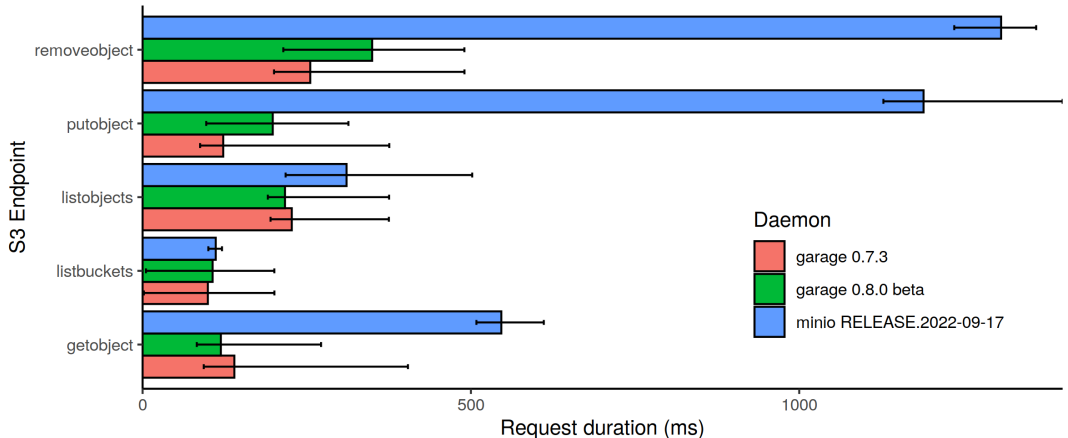
Performance evaluation

S3 endpoint latency in a simulated geo-distributed cluster

100 measurements, 5 nodes, 50ms RTT + 10ms jitter between nodes

no contention: latency is due to intra-cluster communications

colored bar = mean latency, error bar = min and max latency



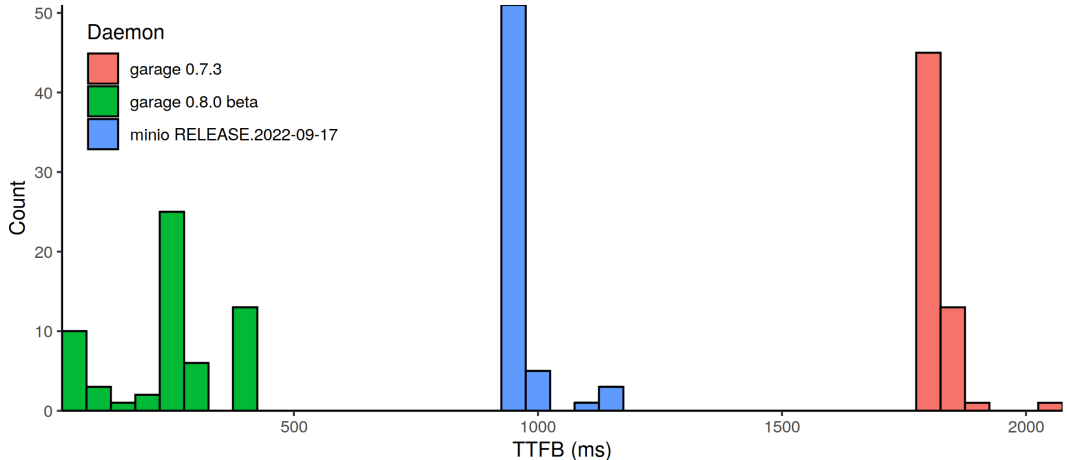
Get the code to reproduce this graph at <https://git.deuxfleurs.fr/Deuxfleurs/mknet>

Performance evaluation

TTFB (Time To First Byte) on GetObject over a slow network (5 Mbps, 500 μ s)

A 1MB file is uploaded and then fetched 60 times.

Except for Minio, the queried node does not store any data (gateway) to force net. communications.



Get the code to reproduce this graph at <https://git.deuxfleurs.fr/Deuxfleurs/mknet>

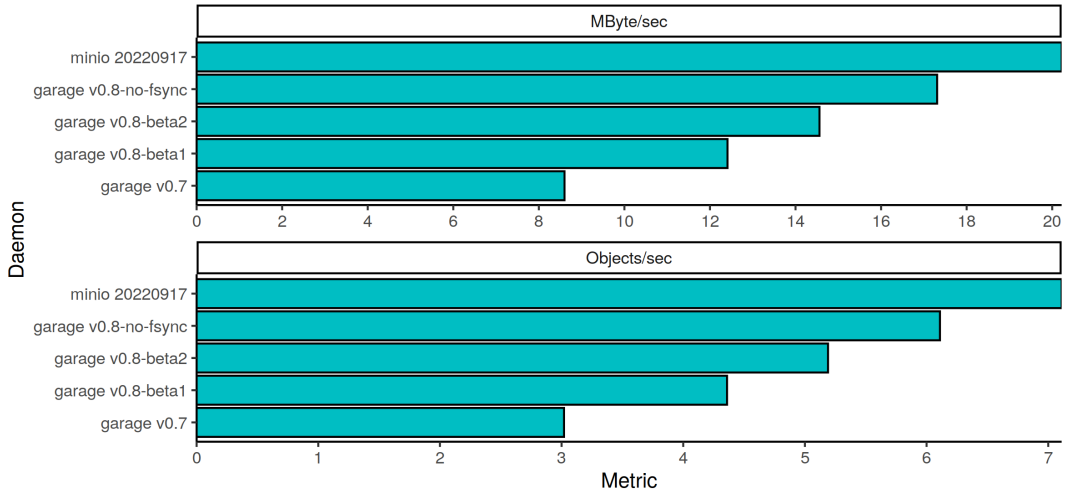
Performance evaluation

"minio/warp" benchmark, "cluster total" result

Ran on a local machine (Ryzen 5 1400, 16GB RAM, SSD) with mknet

DC topology (3 nodes, 1GB/s, 1ms lat)

warp in mixed mode, 5min bench, 5MB objects, initialized with 200 objects

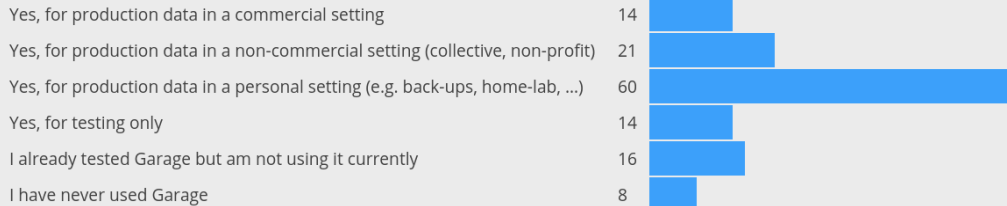


Get the code to reproduce this graph at <https://git.deuxfleurs.fr/Deuxfleurs/mknet>

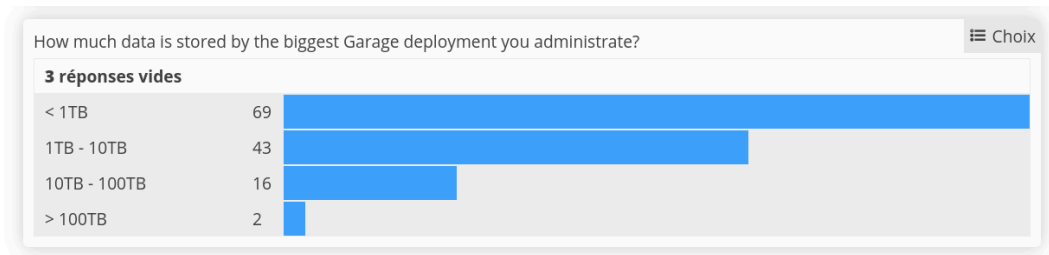
Garage in the wild

Are you currently an administrator of a Garage deployment?

☰ Choix



Size of known deployments



“Petabyte storage setup for a video site. Nginx as CDN in-front using garage-s3-website feature. Each storage node has 64TB storage with raid10, no replication within garage. 25gbit nic. haproxy to loadbalance across 5 nodes. mostly reads with very few writes.”

“We currently manage 7 Garage nodes, 28TB total storage, 6M blocks for 3M objects and 4TB of object data. We have been running Garage in production for 2.5 years.”

Deploying Garage

Choosing a replication factor

Replication factor	Pro	Cons
1	easy single-node setup full space efficiency	no metadata redundancy vulnerable to hardware crash or data corruption no high-availability
2	redundancy limited storage overhead	limited high-availability (read-only when one node is unavailable)
3	high-availability setup best data resilience	big storage overhead
4, 5, ...	possible if needed	...

Important note: metadata replication == data replication
Choose well, this cannot be changed easily!

Setting up data and metadata storage

	Metadata storage	Data storage
Content	access keys, buckets index of objects	raw data blocks
Size	< 10% of data rarely over 100GB	replication × dataset size no erasure-coding
Constraints	latency sensitive write-intensive under load	big many files
Ideal hardware	enterprise-grade SSD	HDD
Recommended redundancy	RAID1	none, use disks directly avoid RAID if possible
Recommended filesystem	ZFS, Btrfs	XFS on individual disks
Tunables in Garage	database engine automatic snapshots	block size compression

Picking a metadata engine

All files-to-block mappings are stored in the metadata engine, including bucket and object metadata. Files below 3KB are stored directly in the metadata engine.

Metadata engine	Characteristics	Use case
SQLite	safer	single node deployment small clusters clusters with infrequent access
LMDB	faster sometimes has inexplicable corruptions	larger clusters with metadata redundancy
Fjall	experimental best of both worlds?	help us test it!

Metadata engine can be set node per-node, and changed later with a migration tool

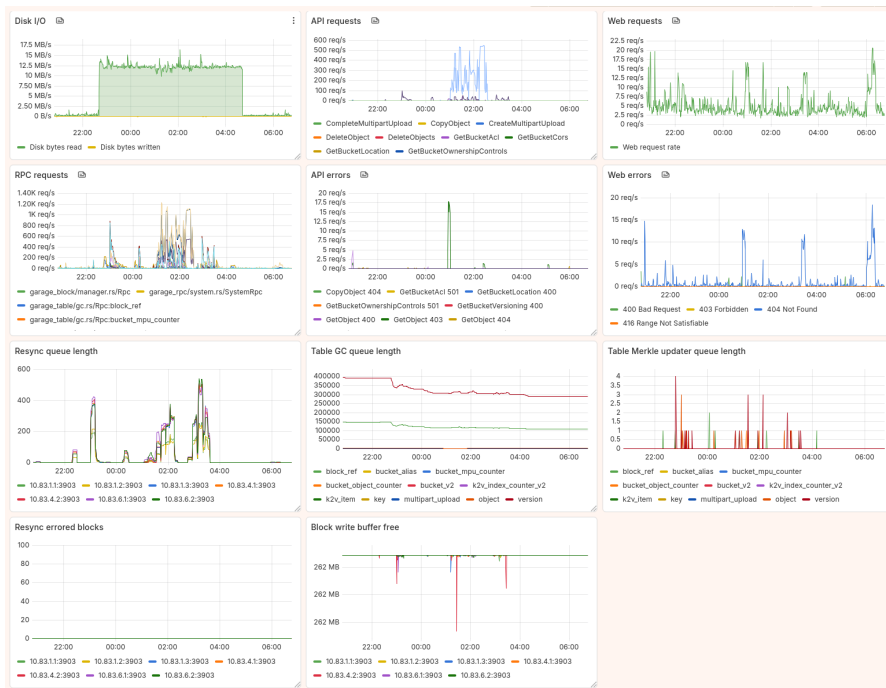
Avoiding common issues as soon as possible

Risk	How to avoid
Metadata corruption (esp. with LMDB)	Configure automatic snapshots with <code>metadata_auto_snapshot_interval</code> Use replication factor 2 or 3
Data not well balanced between nodes	Avoid clusters with too many nodes Target: $\#nodes \leq 10 \times replication_factor$
Performance issues with many objects in one single bucket	Spread your data over multiple buckets
Performance issues with big objects	Increase <code>block_size</code> configuration parameter Target: $object\ size \leq 1000 \times block_size$, $block_size \leq 100MB$
Performance issues with many small objects	Have enough RAM to fit the entire metadata DB

Other things to consider during set-up

Tools for cluster deployment	Ansible + systemd NixOS Kubernetes or Nomad with Docker
Initial cluster setup	Manual layout configuration Read the documentation!
TLS support on public endpoints	Add an external reverse-proxy (Nginx, ...)
S3 anonymous access	Not implemented, use website endpoint
Monitoring	Prometheus + Grafana for Garage metrics External tool to monitor HDD health

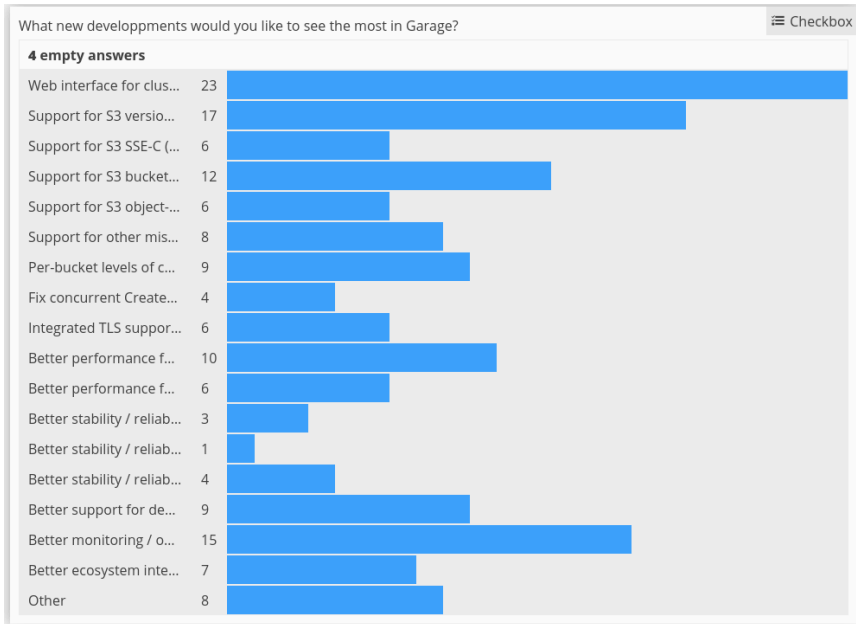
Monitoring with Prometheus + Grafana



Common issues and their solutions

Problem	Solution
S3 access authorization issues	Correctly set the region parameter in your S3 client default = garage, not us-east-1
	Check your reverse proxy configuration
Debugging other API issues	Set RUST_LOG=garage=debug to investigate
Resync queue fills up	garage worker set -a resync-worker-count 8 garage worker set -a resync-tranquility 0
LMDB database too big	Stop garage and compact with <code>mdb_copy -c</code>
Data recovery with dead/ unavailable nodes	Consistency mode degraded allows to read data from an unhealthy cluster. Do not use it for regular operation.
Other issues	Ask us on matrix <code>#garage:deuxfleurs.fr</code> or open an issue on <code>git.deuxfleurs.fr</code> Provide the output of <code>garage status</code> , <code>garage stats</code> and relevant metrics and logs

Future developments



Future developments

The screenshot displays the 'Garage Web Admin' interface. At the top left is the logo and 'Web Admin' text. On the right, there's a status indicator 'Connected node garage-n1' and a 'Help' icon. A left sidebar contains navigation links: Overview (selected), STORAGE, Buckets, ACCESS, Access Keys, Admin Tokens, CLUSTER, Nodes, and Layout. The main content area is titled 'Overview' and includes a 'Refresh' button and 'Last updated 2min ago' text. Below this, a summary row shows: Connected nodes (5/5 known), Storage nodes (4/4, Cluster Healthy, 100% connected), Data space available (671 GB estimated), and Metadata space available (663 GB estimated). Three cards below show 'Buckets' (450 active), 'Access keys' (5/10 active), and 'Admin tokens' (3/9 active). The 'Storage health' section lists three nodes: 'garage-n1' (green dot), 'garage-n2' (orange triangle, last seen 2min ago), and 'garage-n3' (green dot). Each node card shows 'Resync queue: 2255', 'Resync error' (1 triangle), 'Scrub corruption: 0', and 'Disk usage: 45%' with a progress bar. 'Data Disk usage' and 'Metadata Disk usage' are also shown for 'garage-n1' with their respective progress bars and free space (225 GB free).

Garage Web Admin

Connected node garage-n1 Help

Overview Last updated 2min ago Refresh


Connected nodes	Storage nodes	Data space available	Metadata space available
5 / 5 known	4 / 4 Cluster Healthy 100% connected	671 GB estimated	663 GB estimated

[Buckets](#) 450 active → [Access keys](#) 5 / 10 active → [Admin tokens](#) 3 / 9 active →

Storage health All nodes →

garage-n1	Resync queue: 2255 Resync error: 1 ▲ Scrub corruption: 0	Data Disk usage: 45% Metadata Disk usage: 30%	225 GB free 30 GB free →
garage-n2 last seen 2min ago	Resync queue: 2255 Resync error: 0 Scrub corruption: 0	Disk usage: 45%	225 GB free →
garage-n3	Resync queue: 2255 Resync error: 1 ▲ Scrub corruption: 0	Disk usage: 45%	225 GB free →

Future developments

 Buckets / my-bucket / Keys allowed Connected node garage-n1 Help

my-bucket

bucket-alias local, super-bucket-alias local [Go manage aliases](#)

7ac5a87a176f4eba003ea49d6db5aea5fa42eef21bb19c2a51d44968d30ca7f9 Last updated 2min ago Refresh Actions

Creation date	Space	Objects	Keys	Unfinished Uploads	Website access
22 Feb 2024	3 GB 50% of 6 GB	64 065 12% of 500K	8 1 expired	565 (300 MB) with 6436 parts	Enabled

[Keys](#) [Aliases](#) [Website access](#)

Keys allowed

7 displayed Sort

Name / id	Creation date	Permissions	Expiration	Aliases
a-key-name Gkb74ad322b45899a20d4872b8	22 Feb 2025	Read/Write	∞	1
a-key-name Gkb74ad322b45899a20d4872b8	13 Dec 2024	Read/Write/Owner	13 Dec 2024 expired	0
a-key-name Gkb74ad322b45899a20d4872b8	22 Nov 2024	Read/Write	∞	2

Where to find us



Garage

<https://garagehq.deuxfleurs.fr/>
garagehq@deuxfleurs.fr
#garage:deuxfleurs.fr on Matrix

