An introduction to Garage

Alex Auvolat, Deuxfleurs

`https://garagehq.deuxfleurs.fr/`

Matrix channel: `#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

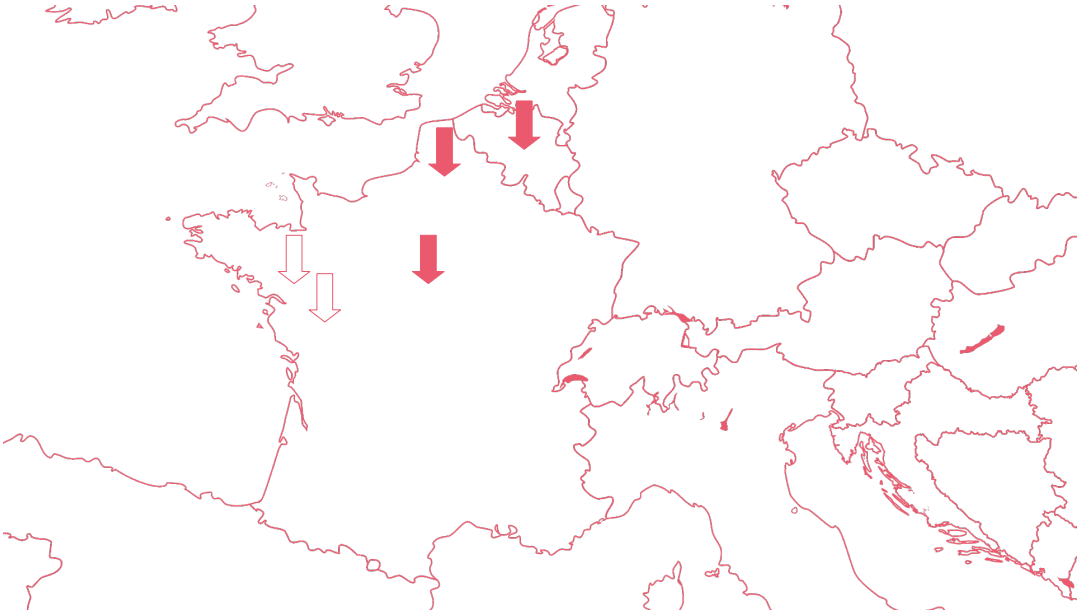
Our very low-tech infrastructure

- 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



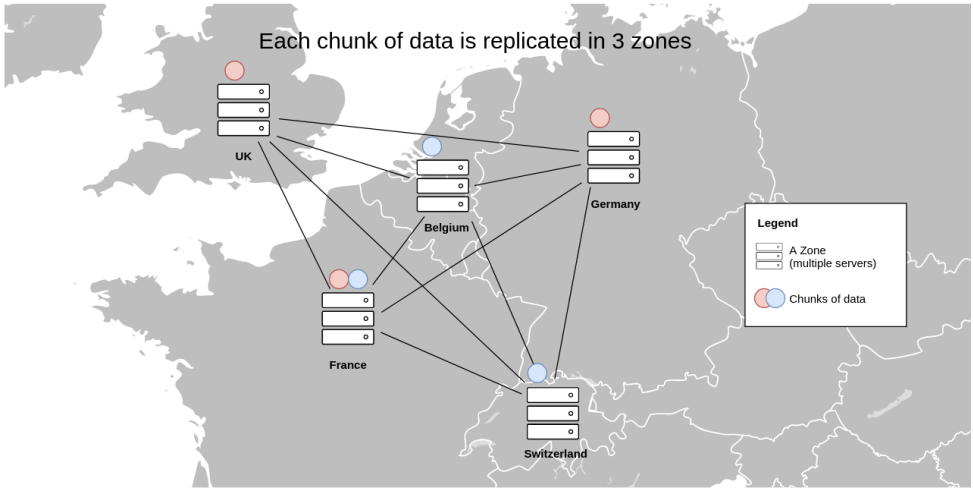
Object storage: a crucial component

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  concombre  [2001:470:ca43::31]:3901                [concombre,neptune,france,alex]         neptune   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>`

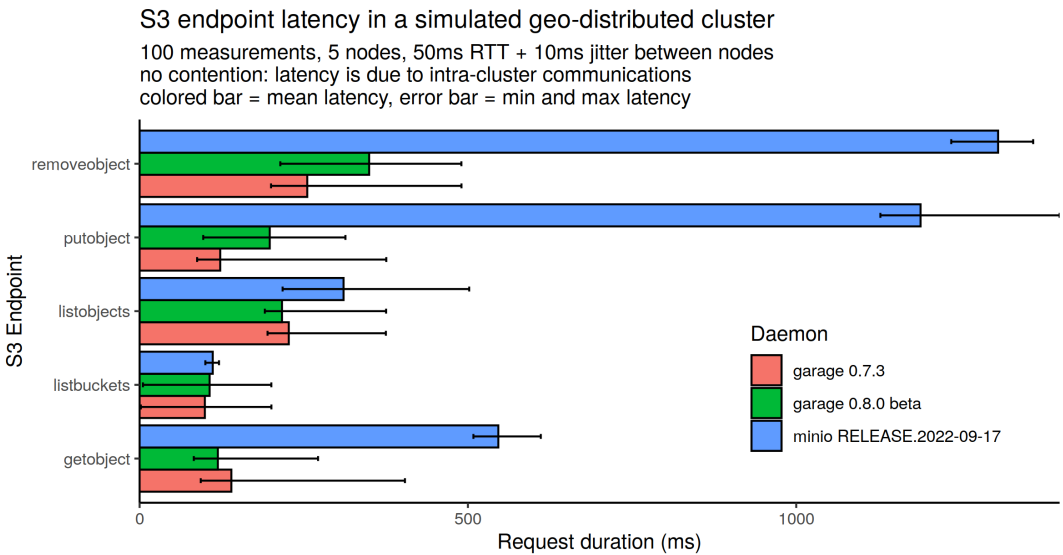
`img/logo.svg`	`Content-Type: text/svg+xml`
	`Content-Length: 13429`
	`<binary blob>`

`download/index.html`	`Content-Type: text/html; charset=utf-8`
	`Content-Length: 26563`
	`<binary blob>`

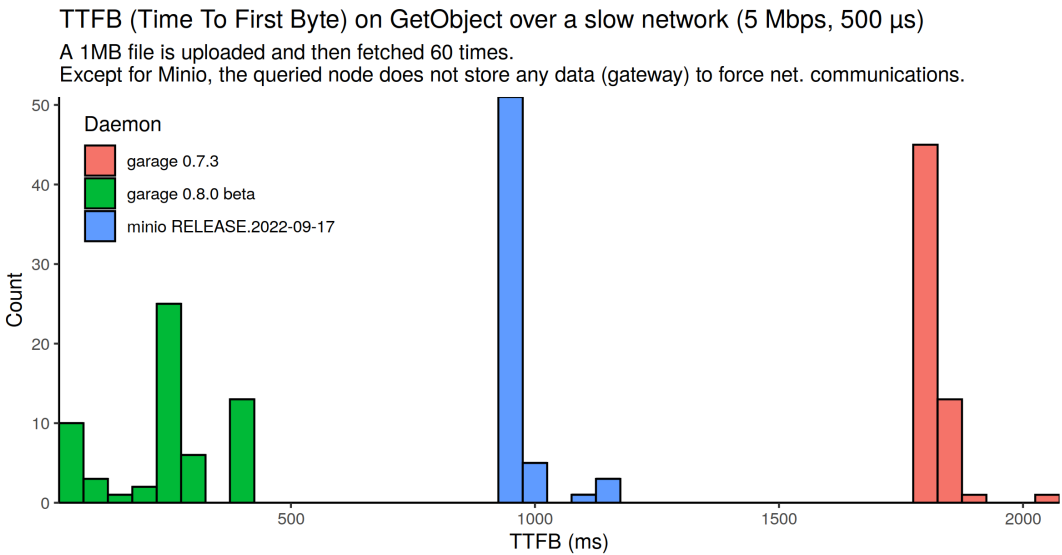
Consistency model:

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

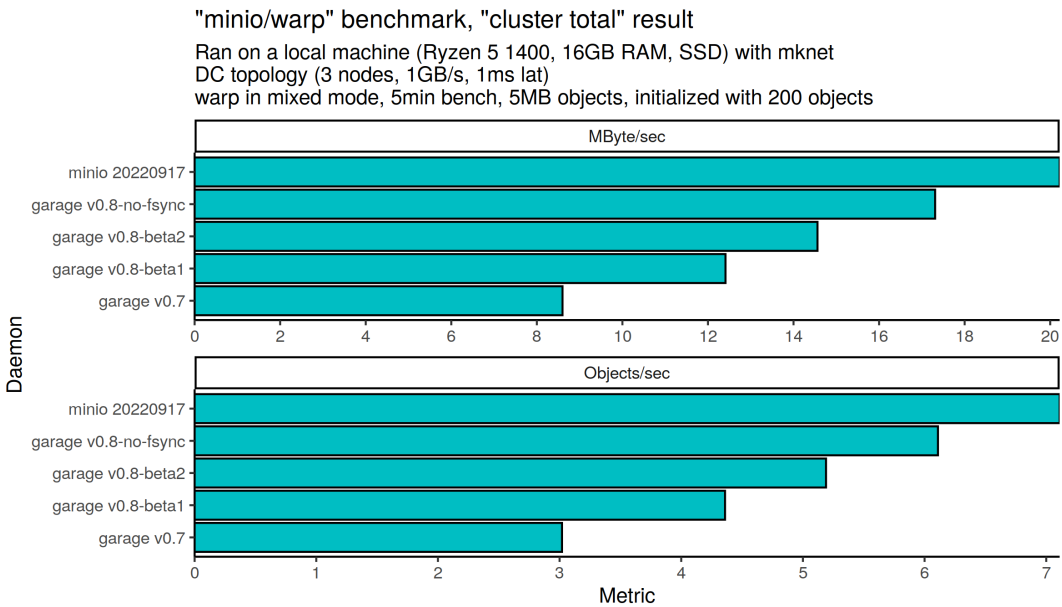
Performance evaluation



Performance evaluation

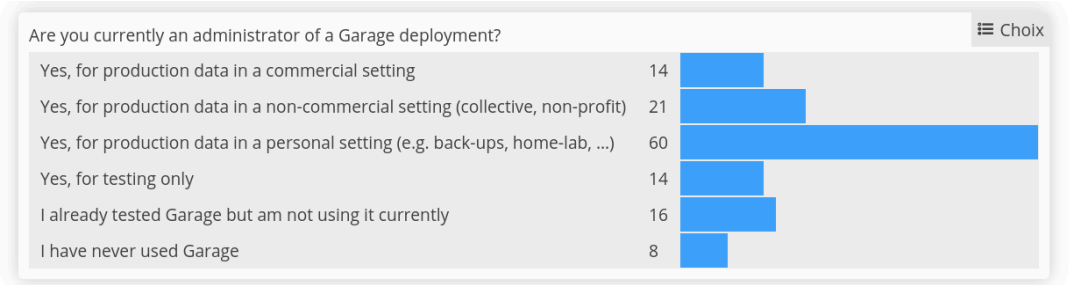


Performance evaluation

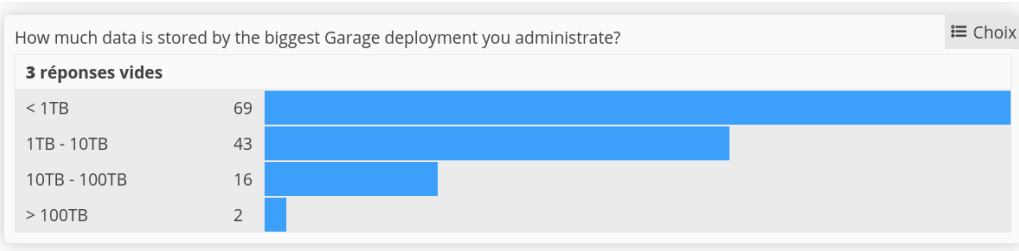


Get the code to reproduce this graph at https://git.deuxfleurs.fr/Deuxfleurs/mknet

Garage in the wild



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

Chosing a replication factor

Replication factor	Pro	Cons
1	easy single-node setup full space efficiency	no metadata redundancy **vunlerable 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	entreprise-grade SSD	HDD
Recommended redundancy	RAID1	none, use disks directly **avoid RAID if possible**
Recommended filesystem	ZFS, Btrfs	XFS on invidual 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

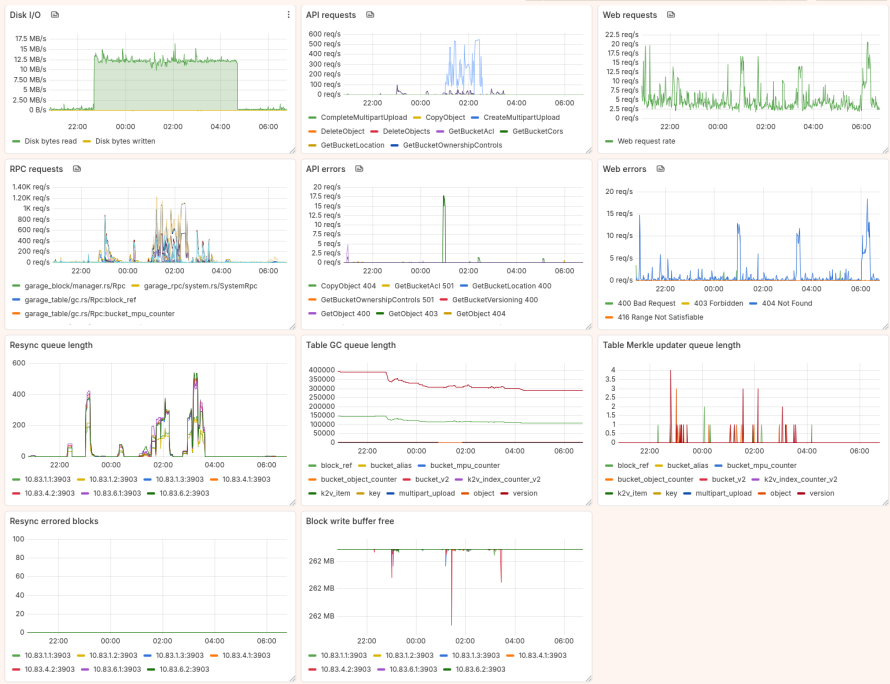
Avoiding common issues as soon as possible

Risk	How to avoid
Metadata corruption (esp. with LMDB)	Configure automatic snapshots with `metadata_auto_snapshot_interval` Use replication factor 2 or 3
Data not well balanced between nodes	Avoid clusters with too many nodes Target: #nodes ≤ 10 × replication_factor
Performance issues with many objects in one single bucket	Spread your data over multiple buckets
Performance issues with big objects	Increase `block_size` configuration parameter Target: object size ≤ 1000 × `block_size`, `block_size` ≤ 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

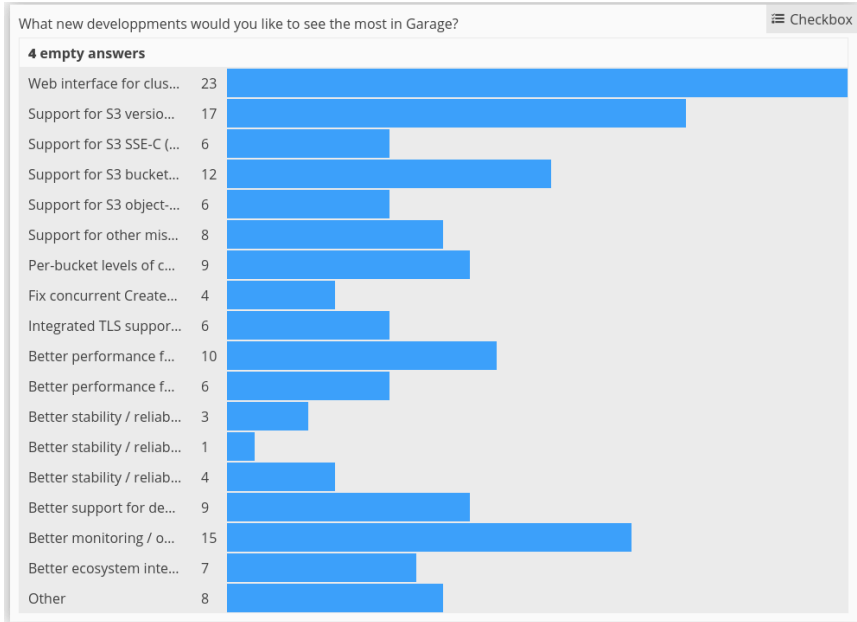
Monitoring with Prometheus + Grafana



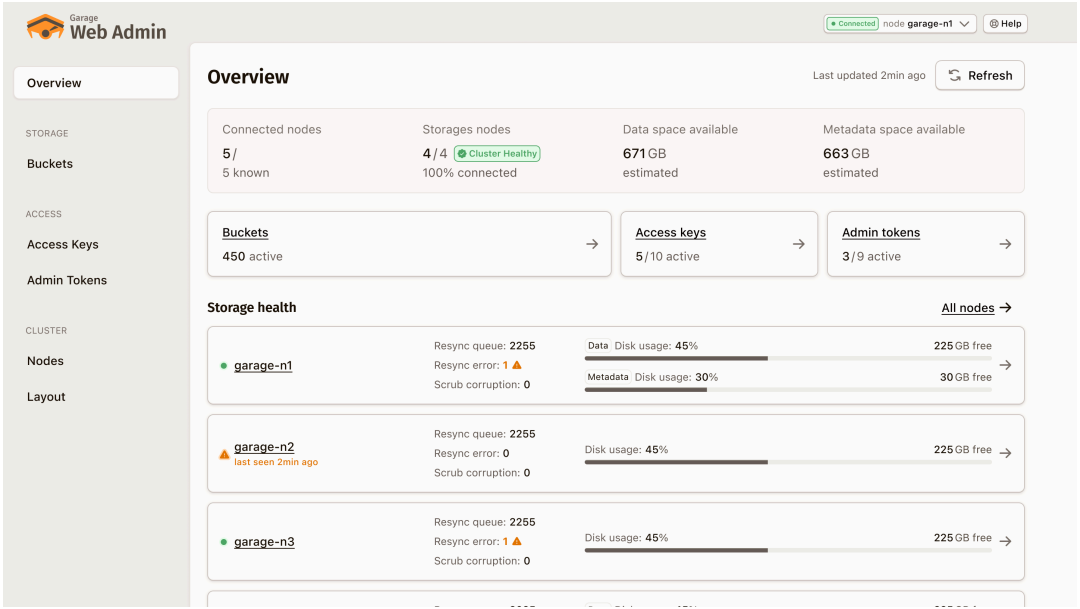
Common issues and their solutions

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 `mdb_copy -c`
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 `#garage:deuxfleurs.fr` or open an issue on `git.deuxfleurs.fr` Provide the output of `garage status`, `garage stats` and relevant metrics and logs

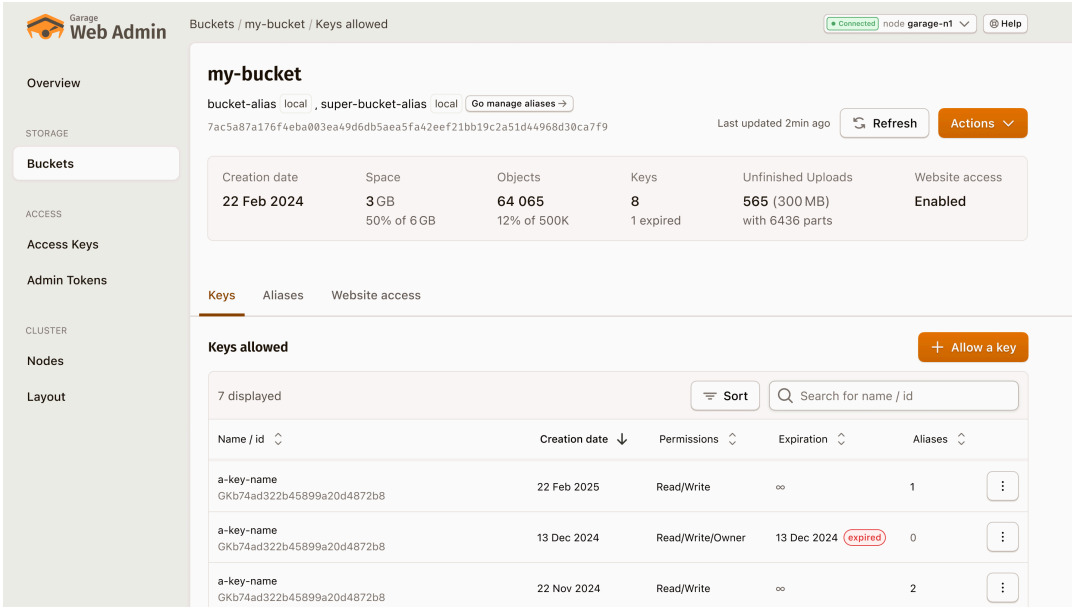
Future developments



Future developments



Future developments



Where to find us

`https://garagehq.deuxfleurs.fr/`

`garagehq@deuxfleurs.fr`

`#garage:deuxfleurs.fr` on Matrix