Garbage collection

Estimated reading time: 4 minutes

As of v2.4.0 a garbage collector command is included within the registry binary. This document describes what this command does and how and why it should be used.

What is Garbage Collection?

From wikipedia:

“In computer science, garbage collection (GC) is a form of automatic memory management. The garbage collector, or just collector, attempts to reclaim garbage, or memory occupied by objects that are no longer in use by the program.”

In the context of the Docker registry, garbage collection is the process of removing blobs from the filesystem which are no longer referenced by a manifest. Blobs can include both layers and manifests.

Why Garbage Collection?

Registry data can occupy considerable amounts of disk space and freeing up this disk space is an oft-requested feature. Additionally for reasons of security it can be desirable to ensure that certain layers no longer exist on the filesystem.

Garbage Collection in the Registry

Filesystem layers are stored by their content address in the Registry. This has many advantages, one of which is that data is stored once and referred to by manifests. See here for more details.

Layers are therefore shared amongst manifests; each manifest maintains a

reference to the layer. As long as a layer is referenced by one manifest, it cannot be garbage collected.

Manifests and layers can be ‘deleted` with the registry API (refer to the API documentation here and here for details). This API removes references to the target and makes them eligible for garbage collection. It also makes them unable to be read via the API.

If a layer is deleted it will be removed from the filesystem when garbage collection is run. If a manifest is deleted the layers to which it refers will be removed from the filesystem if no other manifests refers to them.

Example

In this example manifest A references two layers: `a` and `b`. Manifest `B` references layers `a` and `c`. In this state, nothing is eligible for garbage collection:

```
A -----> a <----- B
    \--> b      |
         c <--/
```

Manifest B is deleted via the API:

```
A -----> a      B
    \--> b
         c
```

In this state layer `c` no longer has a reference and is eligible for garbage collection. Layer `a` had one reference removed but will not be garbage collected as it is still referenced by manifest `A`. The blob representing manifest `B` will also be eligible for garbage collection.

After garbage collection has been run manifest `A` and its blobs remain.

How Garbage Collection works

Garbage collection runs in two phases. First, in the 'mark' phase, the process scans all the manifests in the registry. From these manifests, it constructs a set of content address digests. This set is the 'mark set' and denotes the set of blobs to *not* delete. Secondly, in the 'sweep' phase, the process scans all the blobs and if a blob's content address digest is not in the mark set, the process will delete it.

> **NOTE** You should ensure that the registry is in read-only mode or not running at all. If you were to upload an image while garbage collection is running, there is the risk that the image's layers will be mistakenly deleted, leading to a corrupted image.

This type of garbage collection is known as stop-the-world garbage collection. In future registry versions the intention is that garbage collection will be an automated background action and this manual process will no longer apply.

Running garbage collection

Garbage collection can be run as follows

```
bin/registry garbage-collect [--dry-run] /path/to/config.yml
```

The garbage-collect command accepts a `--dry-run` parameter, which will print the progress of the mark and sweep phases without removing any data. Running with a log level of `info` will give a clear indication of what will and will not be deleted.

Sample output from a dry run garbage collection with registry log level set to `info`

```
hello-world
hello-world: marking manifest sha256:fea8895f450959fa676bcc1df0611ea93
```

```
hello-world: marking blob sha256:03f4658f8b782e12230c1783426bd3bacce65
hello-world: marking blob sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d6
hello-world: marking configuration sha256:690ed74de00f99a7d00a98a5ad85
ubuntu

4 blobs marked, 5 blobs eligible for deletion
blob eligible for deletion: sha256:28e09fddaacbfc8a13f82871d9d66141a6e
blob eligible for deletion: sha256:7e15ce58ccb2181a8fced7709e9893206f0
blob eligible for deletion: sha256:87192bdbe00f8f2a62527f36bb4c7c7f4ea
blob eligible for deletion: sha256:b549a9959a664038fc35c155a95742cf122
blob eligible for deletion: sha256:f251d679a7c61455f06d793e43c06786d77
```

Feedback? Suggestions? Can't find something in the docs?

Edit this page ● Request docs changes ● Get support

Rate this page: