docker image ls

By Docker Inc

Published: 2001-01-01 · Archived: 2026-04-06 01:16:02 UTC

Description	List images
Usage	`docker image ls [OPTIONS] [REPOSITORY[:TAG]]`
Aliases **An alias is a short or memorable alternative for a longer command.**	`docker image list` `docker images`

The default `docker images` will show all top level images, their repository and tags, and their size.

Docker images have intermediate layers that increase reusability, decrease disk usage, and speed up `docker build` by allowing each step to be cached. These intermediate layers are not shown by default.

Untagged (dangling) images are also hidden by default. Use the `-a` (`--all`) flag to show intermediate layers and dangling images.

The `SIZE` is the cumulative space taken up by the image and all its parent images. This is also the disk space used by the contents of the Tar file created when you `docker save` an image.

An image will be listed more than once if it has multiple repository names or tags. This single image (identifiable by its matching `IMAGE ID`) uses up the `SIZE` listed only once.

Option	Default	Description
`-a, --all`		Show all images (default hides intermediate and dangling images)
`--digests`		Show digests
`-f, --filter`		Filter output based on conditions provided
`--format`		Format output using a custom template: 'table': Print output in table format with column headers (default) 'table TEMPLATE': Print output in table format using the given Go template 'json': Print in JSON format 'TEMPLATE': Print output using the given Go template.

Option	Default	Description
		Refer to https://docs.docker.com/go/formatting/ for more information about formatting output with templates
`--no-trunc`		Don't truncate output
`-q, --quiet`		Only show image IDs
`--tree`		API 1.47+ experimental (CLI) List multi-platform images as a tree (EXPERIMENTAL)

List the most recently created images

List images by name and tag

The `docker images` command takes an optional `[REPOSITORY[:TAG]]` argument that restricts the list to images that match the argument. If you specify `REPOSITORY` but no `TAG`, the `docker images` command lists all images in the given repository.

For example, to list all images in the `java` repository, run the following command:

The `[REPOSITORY[:TAG]]` value must be an exact match. This means that, for example, `docker images jav` does not match the image `java`.

If both `REPOSITORY` and `TAG` are provided, only images matching that repository and tag are listed. To find all local images in the `java` repository with tag `8` you can use:

If nothing matches `REPOSITORY[:TAG]`, the list is empty.

List the full length image IDs (--no-trunc)

List image digests (--digests)

Images that use the v2 or later format have a content-addressable identifier called a `digest`. As long as the input used to generate the image is unchanged, the digest value is predictable. To list image digest values, use the `--digests` flag:

When pushing or pulling to a 2.0 registry, the `push` or `pull` command output includes the image digest. You can `pull` using a digest value. You can also reference by digest in `create`, `run`, and `rmi` commands, as well as the `FROM` image reference in a Dockerfile.

Filtering (--filter)

The filtering flag (`-f` or `--filter`) format is of "key=value". If there is more than one filter, then pass multiple flags (e.g., `--filter "foo=bar" --filter "bif=baz"`).

The currently supported filters are:

- dangling (boolean - true or false)
- label (`label=<key>` or `label=<key>=<value>`)
- before (`<image-name>[:<tag>]` , `<image id>` or `<image@digest>`) - filter images created before given id or references
- since (`<image-name>[:<tag>]` , `<image id>` or `<image@digest>`) - filter images created since given id or references
- reference (pattern of an image reference) - filter images whose reference matches the specified pattern

Show untagged images (dangling)

This will display untagged images that are the leaves of the images tree (not intermediary layers). These images occur when a new build of an image takes the `repo:tag` away from the image ID, leaving it as `<none>:<none>` or untagged. A warning will be issued if trying to remove an image when a container is presently using it. By having this flag it allows for batch cleanup.

You can use this in conjunction with `docker rmi` :

Docker warns you if any containers exist that are using these untagged images.

Show images with a given label

The `label` filter matches images based on the presence of a `label` alone or a `label` and a value.

The following filter matches images with the `com.example.version` label regardless of its value.

The following filter matches images with the `com.example.version` label with the `1.0` value.

In this example, with the `0.1` value, it returns an empty set because no matches were found.

Filter images by time

The `before` filter shows only images created before the image with a given ID or reference. For example, having these images:

Filtering with `before` would give:

Filtering with `since` would give:

Filter images by reference

The `reference` filter shows only images whose reference matches the specified pattern.

Filtering with `reference` would give:

Filtering with multiple `reference` would give, either match A or B:

Format the output (--format)

The formatting option (`--format`) will pretty print container output using a Go template.

Valid placeholders for the Go template are listed below:

Placeholder	Description
`.ID`	Image ID
`.Repository`	Image repository
`.Tag`	Image tag
`.Digest`	Image digest
`.CreatedSince`	Elapsed time since the image was created
`.CreatedAt`	Time when the image was created
`.Size`	Image disk size

When using the `--format` option, the `image` command will either output the data exactly as the template declares or, when using the `table` directive, will include column headers as well.

The following example uses a template without headers and outputs the `ID` and `Repository` entries separated by a colon (`:`) for all images:

To list all images with their repository and tag in a table format you can use:

To list all images in JSON format, use the `json` directive:

Source: https://docs.docker.com/engine/reference/commandline/images/