docker

Description The base command for the Docker CLI.

Description

Depending on your Docker system configuration, you may be required to preface each `docker` command with `sudo`. To avoid having to use `sudo` with the `docker` command, your system administrator can create a Unix group called `docker` and add users to it.

For more information about installing Docker or `sudo` configuration, refer to the installation instructions for your operating system.

Display help text

To list the help on any command just execute the command, followed by the `--help` option.

```
$ docker run --help

Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Create and run a new container from an image

Options:
      --add-host value             Add a custom host-to-IP mapping (host:ip) (defau
  -a, --attach value               Attach to STDIN, STDOUT or STDERR (default [])
<...>
```

Environment variables

The following environment variables control the behavior of the `docker` command-line client:

Variable	Description
`DOCKER_API_VERSION`	Override the negotiated API version to use for debugging (e.g. `1.19`)
`DOCKER_CERT_PATH`	Location of your authentication keys. This variable is used both by the `docker` CLI and the `dockerd` daemon
`DOCKER_CONFIG`	The location of your client configuration files.
`DOCKER_CONTEXT`	Name of the `docker context` to use (overrides `DOCKER_HOST` env var and default context set with `docker context use`)
`DOCKER_CUSTOM_HEADERS`	(Experimental) Configure custom HTTP headers to be sent by the client. Headers must be provided as a comma-separated list of `name=value` pairs. This is the equivalent to the `HttpHeaders` field in the configuration file.
`DOCKER_DEFAULT_PLATFORM`	Default platform for commands that take the `--platform` flag.
`DOCKER_HIDE_LEGACY_COMMANDS`	When set, Docker hides "legacy" top-level commands (such as `docker rm`, and `docker pull`) in `docker help` output, and only `Management commands` per object-type (e.g., `docker container`) are printed. This may become the default in a future release.
`DOCKER_HOST`	Daemon socket to connect to.
`DOCKER_TLS`	Enable TLS for connections made by the `docker` CLI (equivalent of the `--tls` command-line option). Set to a non-empty value to enable TLS. Note that TLS is enabled automatically if any of the other TLS options are set.
`DOCKER_TLS_VERIFY`	When set Docker uses TLS and verifies the remote. This variable is used both by the `docker` CLI and the `dockerd` daemon
`BUILDKIT_PROGRESS`	Set type of progress output (`auto`, `plain`, `tty`, `rawjson`) when building with BuildKit backend. Use plain to show container output (default `auto`).

Variable	Description
`NO_COLOR`	Disable any ANSI escape codes in the output in accordance with https://no-color.org/

Because Docker is developed using Go, you can also use any environment variables used by the Go runtime. In particular, you may find these useful:

Variable	Description
`HTTP_PROXY`	Proxy URL for HTTP requests unless overridden by NoProxy.
`HTTPS_PROXY`	Proxy URL for HTTPS requests unless overridden by NoProxy.
`NO_PROXY`	Comma-separated values specifying hosts that should be excluded from proxying.

See the Go specification for details on these variables.

Option types

Single character command line options can be combined, so rather than typing `docker run -i -t --name test busybox sh`, you can write `docker run -it --name test busybox sh`.

Boolean

Boolean options take the form `-d=false`. The value you see in the help text is the default value which is set if you do **not** specify that flag. If you specify a Boolean flag without a value, this will set the flag to `true`, irrespective of the default value.

For example, running `docker run -d` will set the value to `true`, so your container **will** run in "detached" mode, in the background.

Options which default to `true` (e.g., `docker build --rm=true`) can only be set to the non-default value by explicitly setting them to `false`:

```
$ docker build --rm=false .
```

Multi

You can specify options like `-a=[]` multiple times in a single command line, for example in these commands:

```
$ docker run -a stdin -a stdout -i -t ubuntu /bin/bash

$ docker run -a stdin -a stdout -a stderr ubuntu /bin/ls
```

Sometimes, multiple options can call for a more complex value string as for `-v`:

```
$ docker run -v /host:/container example/mysql
```

> **Note**
>
> Do not use the `-t` and `-a stderr` options together due to limitations in the `pty` implementation. All `stderr` in `pty` mode simply goes to `stdout`.

Strings and Integers

Options like `--name=""` expect a string, and they can only be specified once. Options like `-c=0` expect an integer, and they can only be specified once.

Configuration files

By default, the Docker command line stores its configuration files in a directory called `.docker` within your `$HOME` directory.

Docker manages most of the files in the configuration directory and you shouldn't modify them. However, you can modify the `config.json` file to control certain aspects of how the `docker` command behaves.

You can modify the `docker` command behavior using environment variables or command-line options. You can also use options within `config.json` to modify some of the same behavior. If an environment variable and the `--config` flag are set, the flag takes precedent over the environment variable. Command line options override environment variables and environment variables override properties you specify in a `config.json` file.

Change the `.docker` directory

To specify a different directory, use the `DOCKER_CONFIG` environment variable or the `--config` command line option. If both are specified, then the `--config` option overrides the `DOCKER_CONFIG` environment variable. The example below overrides the `docker ps` command using a `config.json` file located in the `~/testconfigs/` directory.

```
$ docker --config ~/testconfigs/ ps
```

This flag only applies to whatever command is being ran. For persistent configuration, you can set the `DOCKER_CONFIG` environment variable in your shell (e.g. `~/.profile` or `~/.bashrc`). The example below sets the new directory to be `HOME/newdir/.docker`.

```
$ echo export DOCKER_CONFIG=$HOME/newdir/.docker > ~/.profile
```

Docker CLI configuration file (`config.json`) properties

Use the Docker CLI configuration to customize settings for the `docker` CLI. The configuration file uses JSON formatting, and properties:

By default, configuration file is stored in `~/.docker/config.json`. Refer to the change the `.docker` directory section to use a different location.

> **Warning**
>
> The configuration file and other files inside the `~/.docker` configuration directory may contain sensitive information, such as authentication information for proxies or, depending on your credential store, credentials for your image registries. Review your

configuration file's content before sharing with others, and prevent committing the file to version control.

Customize the default output format for commands

These fields lets you customize the default output format for some commands if no `--format` flag is provided.

Property	Description
`configFormat`	Custom default format for `docker config ls` output. See `docker config ls` for a list of supported formatting directives.
`imagesFormat`	Custom default format for `docker images` / `docker image ls` output. See `docker images` for a list of supported formatting directives.
`networksFormat`	Custom default format for `docker network ls` output. See `docker network ls` for a list of supported formatting directives.
`nodesFormat`	Custom default format for `docker node ls` output. See `docker node ls` for a list of supported formatting directives.
`pluginsFormat`	Custom default format for `docker plugin ls` output. See `docker plugin ls` for a list of supported formatting directives.
`psFormat`	Custom default format for `docker ps` / `docker container ps` output. See `docker ps` for a list of supported formatting directives.
`secretFormat`	Custom default format for `docker secret ls` output. See `docker secret ls` for a list of supported formatting directives.
`serviceInspectFormat`	Custom default format for `docker service inspect` output. See `docker service inspect` for a list of supported formatting directives.
`servicesFormat`	Custom default format for `docker service ls` output. See `docker service ls` for a list of supported formatting directives.
`statsFormat`	Custom default format for `docker stats` output. See `docker stats` for a list of supported formatting directives.

Property	Description
`tasksFormat`	Custom default format for `docker stack ps` output. See `docker stack ps` for a list of supported formatting directives.
`volumesFormat`	Custom default format for `docker volume ls` output. See `docker volume ls` for a list of supported formatting directives.

Custom HTTP headers

The property `HttpHeaders` specifies a set of headers to include in all messages sent from the Docker client to the daemon. Docker doesn't try to interpret or understand these headers; it simply puts them into the messages. Docker does not allow these headers to change any headers it sets for itself.

Alternatively, use the `DOCKER_CUSTOM_HEADERS` environment variable, which is available in v27.1 and higher. This environment-variable is experimental, and its exact behavior may change.

Credential store options

The property `credsStore` specifies an external binary to serve as the default credential store. When this property is set, `docker login` will attempt to store credentials in the binary specified by `docker-credential-<value>` which is visible on `$PATH`. If this property isn't set, credentials are stored in the `auths` property of the CLI configuration file. For more information, see the **Credential stores** section in the `docker login` documentation

The property `credHelpers` specifies a set of credential helpers to use preferentially over `credsStore` or `auths` when storing and retrieving credentials for specific registries. If this property is set, the binary `docker-credential-<value>` will be used when storing or retrieving credentials for a specific registry. For more information, see the **Credential helpers** section in the `docker login` documentation

Automatic proxy configuration for containers

The property `proxies` specifies proxy environment variables to be automatically set on containers, and set as `--build-arg` on containers used during `docker build`. A `"default"` set of proxies can be configured, and will be used for any Docker daemon that the client connects

to, or a configuration per host (Docker daemon), for example, `https://docker-daemon1.example.com`. The following properties can be set for each environment:

Property	Description
`httpProxy`	Default value of `HTTP_PROXY` and `http_proxy` for containers, and as `--build-arg` on `docker build`
`httpsProxy`	Default value of `HTTPS_PROXY` and `https_proxy` for containers, and as `--build-arg` on `docker build`
`ftpProxy`	Default value of `FTP_PROXY` and `ftp_proxy` for containers, and as `--build-arg` on `docker build`
`noProxy`	Default value of `NO_PROXY` and `no_proxy` for containers, and as `--build-arg` on `docker build`
`allProxy`	Default value of `ALL_PROXY` and `all_proxy` for containers, and as `--build-arg` on `docker build`

These settings are used to configure proxy settings for containers only, and not used as proxy settings for the `docker` CLI or the `dockerd` daemon. Refer to the environment variables section and the Daemon proxy configuration guide for configuring proxy settings for the CLI and daemon.

> ⚠ **Warning**
>
> Proxy settings may contain sensitive information (for example, if the proxy requires authentication). Environment variables are stored as plain text in the container's configuration, and as such can be inspected through the remote API or committed to an image when using `docker commit`.

Default key-sequence to detach from containers

Once attached to a container, users detach from it and leave it running using the using `CTRL-p CTRL-q` key sequence. This detach key sequence is customizable using the `detachKeys`

property. Specify a `<sequence>` value for the property. The format of the `<sequence>` is a comma-separated list of either a letter [a-Z], or the `ctrl-` combined with any of the following:

- `a-z` (a single lowercase alpha character)
- `@` (at sign)
- `[` (left bracket)
- `\\` (two backward slashes)
- `_` (underscore)
- `^` (caret)

Your customization applies to all containers started in with your Docker client. Users can override your custom or the default key sequence on a per-container basis. To do this, the user specifies the `--detach-keys` flag with the `docker attach`, `docker exec`, `docker run` or `docker start` command.

CLI plugin options

The property `plugins` contains settings specific to CLI plugins. The key is the plugin name, while the value is a further map of options, which are specific to that plugin.

Sample configuration file

Following is a sample `config.json` file to illustrate the format used for various fields:

```
{
  "HttpHeaders": {
    "MyHeader": "MyValue"
  },
  "psFormat": "table {{.ID}}\\t{{.Image}}\\t{{.Command}}\\t{{.Labels}}",
  "imagesFormat": "table {{.ID}}\\t{{.Repository}}\\t{{.Tag}}\\t{{.CreatedAt}}",
  "pluginsFormat": "table {{.ID}}\t{{.Name}}\t{{.Enabled}}",
  "statsFormat": "table {{.Container}}\t{{.CPUPerc}}\t{{.MemUsage}}",
  "servicesFormat": "table {{.ID}}\t{{.Name}}\t{{.Mode}}",
  "secretFormat": "table {{.ID}}\t{{.Name}}\t{{.CreatedAt}}\t{{.UpdatedAt}}",
  "configFormat": "table {{.ID}}\t{{.Name}}\t{{.CreatedAt}}\t{{.UpdatedAt}}",
  "serviceInspectFormat": "pretty",
  "nodesFormat": "table {{.ID}}\t{{.Hostname}}\t{{.Availability}}",
```

```
  "detachKeys": "ctrl-e,e",
  "credsStore": "secretservice",
  "credHelpers": {
    "awesomereg.example.org": "hip-star",
    "unicorn.example.com": "vcbait"
  },
  "plugins": {
    "plugin1": {
      "option": "value"
    },
    "plugin2": {
      "anotheroption": "anothervalue",
      "athirdoption": "athirdvalue"
    }
  },
  "proxies": {
    "default": {
      "httpProxy":  "http://user:pass@example.com:3128",
      "httpsProxy": "https://my-proxy.example.com:3129",
      "noProxy":    "intra.mycorp.example.com",
      "ftpProxy":   "http://user:pass@example.com:3128",
      "allProxy":   "socks://example.com:1234"
    },
    "https://manager1.mycorp.example.com:2377": {
      "httpProxy":  "http://user:pass@example.com:3128",
      "httpsProxy": "https://my-proxy.example.com:3129"
    }
  }
}
```

Experimental features

Experimental features provide early access to future product functionality. These features are intended for testing and feedback, and they may change between releases without warning or can be removed from a future release.

Starting with Docker 20.10, experimental CLI features are enabled by default, and require no configuration to enable them.

Notary

If using your own notary server and a self-signed certificate or an internal Certificate Authority, you need to place the certificate at `tls/<registry_url>/ca.crt` in your Docker config directory.

Alternatively you can trust the certificate globally by adding it to your system's list of root Certificate Authorities.

Options

Option	Default	Description
`--config`	`/root/.docker`	Location of client config files
`-c, --context`		Name of the context to use to connect to the daemon (overrides DOCKER_HOST env var and default context set with `docker context use`)
`-D, --debug`		Enable debug mode
`-H, --host`		Daemon socket to connect to
`-l, --log-level`	`info`	Set the logging level (`debug`, `info`, `warn`, `error`, `fatal`)
`--tls`		Use TLS; implied by --tlsverify
`--tlscacert`	`/root/.docker/ca.pem`	Trust certs signed only by this CA
`--tlscert`	`/root/.docker/cert.pem`	Path to TLS certificate file
`--tlskey`	`/root/.docker/key.pem`	Path to TLS key file
`--tlsverify`		Use TLS and verify the remote

Examples

Specify daemon host (-H, --host)

You can use the `-H`, `--host` flag to specify a socket to use when you invoke a `docker` command. You can use the following protocols:

Scheme	Description	Example
`unix://[<path>]`	Unix socket (Linux only)	`unix:///var/run/docker.sock`
`tcp://[<IP or host>[:port]]`	TCP connection	`tcp://174.17.0.1:2376`
`ssh://[username@]<IP or host>[:port]`	SSH connection	`ssh://user@192.168.64.5`
`npipe://[<name>]`	Named pipe (Windows only)	`npipe:////./pipe/docker_engine`

If you don't specify the `-H` flag, and you're not using a custom context, commands use the following default sockets:

- `unix:///var/run/docker.sock` on macOS and Linux
- `npipe:////./pipe/docker_engine` on Windows

To achieve a similar effect without having to specify the `-H` flag for every command, you could also create a context, or alternatively, use the `DOCKER_HOST` environment variable.

For more information about the `-H` flag, see Daemon socket option.

Using TCP sockets

The following example shows how to invoke `docker ps` over TCP, to a remote daemon with IP address `174.17.0.1`, listening on port `2376`:

```
$ docker -H tcp://174.17.0.1:2376 ps
```

> **Note**
>
> By convention, the Docker daemon uses port `2376` for secure TLS connections, and port `2375` for insecure, non-TLS connections.

Using SSH sockets

When you use SSH invoke a command on a remote daemon, the request gets forwarded to the `/var/run/docker.sock` Unix socket on the SSH host.

```
$ docker -H ssh://user@192.168.64.5 ps
```

You can optionally specify the location of the socket by appending a path component to the end of the SSH address.

```
$ docker -H ssh://user@192.168.64.5/var/run/docker.sock ps
```

Subcommands

Command	Description
`docker builder`	Manage builds
`docker buildx`	Docker Buildx
`docker checkpoint`	Manage checkpoints
`docker compose`	Docker Compose
`docker config`	Manage Swarm configs
`docker container`	Manage containers
`docker context`	Manage contexts
`docker debug`	Get a shell into any container or image. An alternative to debugging with `docker exec`.
`docker desktop`	Docker Desktop
`docker image`	Manage images
`docker init`	Creates Docker-related starter files for your project
`docker inspect`	Return low-level information on Docker objects
`docker login`	Authenticate to a registry

Command	Description
`docker logout`	Log out from a registry
`docker manifest`	Manage Docker image manifests and manifest lists
`docker mcp`	Manage MCP servers and clients
`docker model`	Docker Model Runner
`docker network`	Manage networks
`docker node`	Manage Swarm nodes
`docker offload`	Control Docker Offload from the CLI
`docker pass`	Manage your local OS keychain secrets.
`docker plugin`	Manage plugins
`docker sandbox`	Docker Sandbox
`docker scout`	Command line tool for Docker Scout
`docker search`	Search Docker Hub for images
`docker secret`	Manage Swarm secrets
`docker service`	Manage Swarm services
`docker stack`	Manage Swarm stacks
`docker swarm`	Manage Swarm
`docker system`	Manage Docker
`docker trust`	Manage trust on Docker images
`docker version`	Show the Docker version information
`docker volume`	Manage volumes