docker login

Estimated reading time: 6 minutes

Description

Log in to a Docker registry

Usage

```
docker login [OPTIONS] [SERVER]
```

Options

Name, shorthand	Default	Description
`--password , -p`		Password
`--password-stdin`		Take the password from stdin
`--username , -u`		Username

Parent command

Command	Description
docker (https://docs.docker.com/engine/reference/commandline/docker)	The base command for the Docker CLI.

Extended description

Login to a registry.

Login to a self-hosted registry

If you want to login to a self-hosted registry you can specify this by adding the server

name.

```
$ docker login localhost:8080
```

Provide a password using STDIN

To run the `docker login` command non-interactively, you can set the `--password-stdin` flag to provide a password through `STDIN`. Using `STDIN` prevents the password from ending up in the shell's history, or log-files.

The following example reads a password from a file, and passes it to the `docker login` command using `STDIN`:

```
$ cat ~/my_password.txt | docker login --username foo --password-stdin
```

Privileged user requirement

`docker login` requires user to use `sudo` or be `root`, except when:

1. connecting to a remote daemon, such as a `docker-machine` provisioned `docker engine`.
2. user is added to the `docker` group. This will impact the security of your system; the `docker` group is `root` equivalent. See Docker Daemon Attack Surface (https://docs.docker.com/engine/security/security/#docker-daemon-attack-surface) for details.

You can log into any public or private repository for which you have credentials. When you log in, the command stores credentials in `$HOME/.docker/config.json` on Linux or `%USERPROFILE%/.docker/config.json` on Windows, via the procedure described below.

Credentials store

The Docker Engine can keep user credentials in an external credentials store, such as the native keychain of the operating system. Using an external store is more secure than storing credentials in the Docker configuration file.

To use a credentials store, you need an external helper program to interact with a specific keychain or external store. Docker requires the helper program to be in the client's host `$PATH`.

This is the list of currently available credentials helpers and where you can download them from:

- D-Bus Secret Service: https://github.com/docker/docker-credential-helpers/releases

- Apple macOS keychain: https://github.com/docker/docker-credential-helpers/releases
- Microsoft Windows Credential Manager: https://github.com/docker/docker-credential-helpers/releases
- pass (https://www.passwordstore.org/): https://github.com/docker/docker-credential-helpers/releases

CONFIGURE THE CREDENTIALS STORE

You need to specify the credentials store in `$HOME/.docker/config.json` to tell the docker engine to use it. The value of the config property should be the suffix of the program to use (i.e. everything after `docker-credential-`). For example, to use `docker-credential-osxkeychain`:

```
{
        "credsStore": "osxkeychain"
}
```

If you are currently logged in, run `docker logout` to remove the credentials from the file and run `docker login` again.

DEFAULT BEHAVIOR

By default, Docker looks for the native binary on each of the platforms, i.e. “osxkeychain” on macOS, “wincred” on windows, and “pass” on Linux. A special case is that on Linux, Docker will fall back to the “secretservice” binary if it cannot find the “pass” binary. If none of these binaries are present, it stores the credentials (i.e. password) in base64 encoding in the config files described above.

CREDENTIAL HELPER PROTOCOL

Credential helpers can be any program or script that follows a very simple protocol. This protocol is heavily inspired by Git, but it differs in the information shared.

The helpers always use the first argument in the command to identify the action. There are only three possible values for that argument: `store`, `get`, and `erase`.

The `store` command takes a JSON payload from the standard input. That payload carries the server address, to identify the credential, the user name, and either a password or an identity token.

```
{
        "ServerURL": "https://index.docker.io/v1",
        "Username": "david",
        "Secret": "passw0rd1"
}
```

If the secret being stored is an identity token, the Username should be set to

`<token>` .

The `store` command can write error messages to `STDOUT` that the docker engine will show if there was an issue.

The `get` command takes a string payload from the standard input. That payload carries the server address that the docker engine needs credentials for. This is an example of that payload: `https://index.docker.io/v1` .

The `get` command writes a JSON payload to `STDOUT` . Docker reads the user name and password from this payload:

```
{
        "Username": "david",
        "Secret": "passw0rd1"
}
```

The `erase` command takes a string payload from `STDIN` . That payload carries the server address that the docker engine wants to remove credentials for. This is an example of that payload: `https://index.docker.io/v1` .

The `erase` command can write error messages to `STDOUT` that the docker engine will show if there was an issue.

Credential helpers

Credential helpers are similar to the credential store above, but act as the designated programs to handle credentials for *specific registries*. The default credential store (`credsStore` or the config file itself) will not be used for operations concerning credentials of the specified registries.

CONFIGURE CREDENTIAL HELPERS

If you are currently logged in, run `docker logout` to remove the credentials from the default store.

Credential helpers are specified in a similar way to `credsStore` , but allow for multiple helpers to be configured at a time. Keys specify the registry domain, and values specify the suffix of the program to use (i.e. everything after `docker-credential-`). For example:

```
{
  "credHelpers": {
    "registry.example.com": "registryhelper",
    "awesomereg.example.org": "hip-star",
    "unicorn.example.io": "vcbait"
  }
}
```