(/)

OFFICIAL REPOSITORY

vault (/r/_/vault/) ☆

Last pushed: 5 days ago

Repo Info (/_/vault/)

Short Description

Vault is a tool for securely accessing secrets via a unified interface and tight access control.

Full Description

Supported tags and respective `Dockerfile` links

- `0.11.5`, `latest` (*0.X/Dockerfile*) (https://github.com/hashicorp/docker-vault/blob/c8425909ddf3f51d46c9dc3d0606562a91fc7dcd/0.X/Dockerfile)
- `1.0.0-beta2` (*0.X/Dockerfile*) (https://github.com/hashicorp/docker-vault/blob/88d41df2e0792ee509b2a8745bb808229d9bee4d/0.X/Dockerfile)

Quick reference

- **Where to get help**:
  the Docker Community Forums (https://forums.docker.com/), the Docker Community Slack (https://blog.docker.com/2016/11/introducing-docker-community-directory-docker-community-slack/), or Stack Overflow (https://stackoverflow.com/search?tab=newest&q=docker)

- **Where to file issues**:
  https://github.com/hashicorp/docker-vault/issues (https://github.com/hashicorp/docker-vault/issues)

- **Maintained by**:
  HashiCorp (https://github.com/hashicorp/docker-vault)

- **Supported architectures**: (more info (https://github.com/docker-library/official-images#architectures-other-than-amd64))
  `amd64` (https://hub.docker.com/r/amd64/vault/), `arm32v6` (https://hub.docker.com/r/arm32v6/vault/), `arm64v8` (https://hub.docker.com/r/arm64v8/vault/), `i386` (https://hub.docker.com/r/i386/vault/)

- **Published image artifact details**:
  repo-info repo's `repos/vault/` directory (https://github.com/docker-library/repo-info/blob/master/repos/vault) (history (https://github.com/docker-library/repo-info/commits/master/repos/vault))
  (image metadata, transfer size, etc)

- **Image updates**:
  official-images PRs with label `library/vault` (https://github.com/docker-library/official-images/pulls?q=label%3Alibrary%2Fvault)
  official-images repo's `library/vault` file (https://github.com/docker-library/official-images/blob/master/library/vault) (history (https://github.com/docker-library/official-images/commits/master/library/vault))

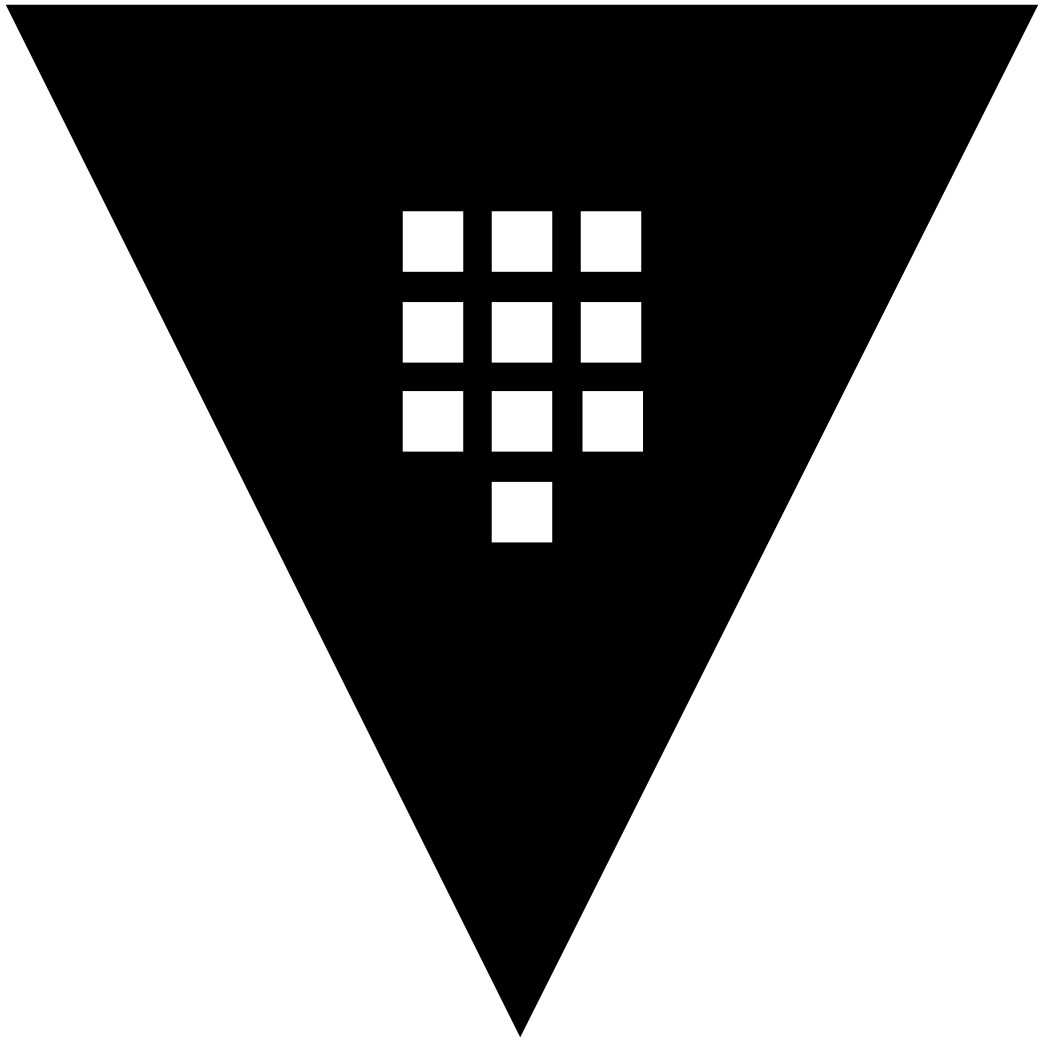
- **Source of this description**:
  docs repo's `vault/` directory (https://github.com/docker-library/docs/tree/master/vault) (history (https://github.com/docker-library/docs/commits/master/vault))

- **Supported Docker versions**:
  the latest release (https://github.com/docker/docker-ce/releases/latest) (down to 1.6 on a best-effort basis)

Vault

Vault is a tool for securely accessing secrets. A secret is anything that you want to tightly control access to, such as API keys, passwords, certificates, and more. Vault provides a unified interface to any secret, while providing tight access control and recording a detailed audit log. For more information, please see:

- Vault documentation (https://www.vaultproject.io/)
- Vault on GitHub (https://github.com/hashicorp/vault)



Using the Container

We chose Alpine as a lightweight base with a reasonably small surface area for security concerns, but with enough functionality for development and interactive debugging.

Vault always runs under dumb-init (https://github.com/Yelp/dumb-init), which handles reaping zombie processes and forwards signals on to all processes running in the container. This binary is built by HashiCorp and signed with our GPG key (https://www.hashicorp.com/security.html), so you can verify the signed package used to build a given base image.

Running the Vault container with no arguments will give you a Vault server in development mode (https://www.vaultproject.io/docs/concepts/dev-server.html). The provided entry point script will also look for Vault subcommands and run `vault` with that subcommand. For example, you can execute `docker run vault status` and it will run the `vault status` command inside the container. The entry point also adds some special

configuration options as detailed in the sections below when running the `server` subcommand. Any other command gets `exec` -ed inside the container under `dumb-init` .

The container exposes two optional `VOLUME` s:

- `/vault/logs` , to use for writing persistent audit logs. By default nothing is written here; the `file` audit backend must be enabled with a path under this directory.
- `/vault/file` , to use for writing persistent storage data when using the `file` data storage plugin. By default nothing is written here (a `dev` server uses an in-memory data store); the `file` data storage backend must be enabled in Vault's configuration before the container is started.

The container has a Vault configuration directory set up at `/vault/config` and the server will load any HCL or JSON configuration files placed here by binding a volume or by composing a new image and adding files. Alternatively, configuration can be added by passing the configuration JSON via environment variable `VAULT_LOCAL_CONFIG` . Please note that due to a bug in the current release of Vault (0.6.0), you should *not* use the name `local.json` for any configuration file in this directory.

Memory Locking and 'setcap'

The container will attempt to lock memory to prevent sensitive values from being swapped to disk and as a result must have `--cap-add=IPC_LOCK` provided to `docker run` . Since the Vault binary runs as a non-root user, `setcap` is used to give the binary the ability to lock memory. With some Docker storage plugins in some distributions this call will not work correctly; it seems to fail most often with AUFS. The memory locking behavior can be disabled by setting the `SKIP_SETCAP` environment variable to any non-empty value.

Running Vault for Development

```
$ docker run --cap-add=IPC_LOCK -d --name=dev-vault vault
```

This runs a completely in-memory Vault server, which is useful for development but should not be used in production.

When running in development mode, two additional options can be set via environment variables:

- `VAULT_DEV_ROOT_TOKEN_ID` : This sets the ID of the initial generated root token to the given value
- `VAULT_DEV_LISTEN_ADDRESS` : This sets the IP:port of the development server listener (defaults to 0.0.0.0:8200)

As an example:

```
$ docker run --cap-add=IPC_LOCK -e 'VAULT_DEV_ROOT_TOKEN_ID=myroot'
```

Running Vault in Server Mode

```
$ docker run --cap-add=IPC_LOCK -e 'VAULT_LOCAL_CONFIG={"backend":
```

This runs a Vault server using the `file` storage backend at path `/vault/file`, with a default secret lease duration of one week and a maximum of 30 days.

Note the `--cap-add=IPC_LOCK`: this is required in order for Vault to lock memory, which prevents it from being swapped to disk. This is highly recommended. In a non-development environment, if you do not wish to use this functionality, you must add `"disable_mlock: true"` to the configuration information.

At startup, the server will read configuration HCL and JSON files from `/vault/config` (any information passed into `VAULT_LOCAL_CONFIG` is written into `local.json` in this directory and read as part of reading the directory for configuration files). Please see Vault's configuration documentation (https://www.vaultproject.io/docs/config/index.html) for a full list of options.

Since 0.6.3 this container also supports the `VAULT_REDIRECT_INTERFACE` and `VAULT_CLUSTER_INTERFACE` environment variables. If set, the IP addresses used for the redirect and cluster addresses in Vault's configuration will be the address of the named interface inside the container (e.g. `eth0`).

License

View license information (https://raw.githubusercontent.com/hashicorp/vault/master/LICENSE) for the software contained in this image.

As with all Docker images, these likely also contain other software which may be under other licenses (such as Bash, etc from the base distribution, along with any direct or indirect dependencies of the primary software being contained).

Some additional license information which was able to be auto-detected might be found in the `repo-info` repository's `vault/` directory (https://github.com/docker-library/repo-info/tree/master/repos/vault).

As for any pre-built image usage, it is the image user's responsibility to ensure that any use of this image complies with any relevant licenses for all software contained within.

Docker Pull Command

```
docker pull vault
```