Hashicorp Vault 1.6 Quick Reference

read	Read data and retrieves secrets	
	vault **read** [options] **PATH**	Reads data from Vault at the given PATH
	vault read secret/my-secret	
write	Write data, configuration, and secrets	
	vault **write** [options] **PATH K=V | K=- | K=@file**	Writes data (key/value pairs) to Vault at the given PATH.
	vault write secret/my-secret foo=bar tom=jerry	If value begins with @ it is loaded from a file
	vault write -f transit/keys/my-key	If value is - it is read from stdin
	vault write aws/roles/ops policy=@policy.json	specify -force/-f if there is no data to write
	echo $MY_TOKEN | vault write consul/config/access token=-	
delete	Delete secrets and configuration	
	vault **delete** [options] **PATH**	Deletes secrets and configuration at PATH.
	vault delete secret/my-secret	The "delete" behaviour is delegated to backend corresponding to the
	vault delete transit/keys/my-key	given path.
	vault delete aws/roles/ops	
list	List data or secrets	
	vault **list** [options] **PATH**	Lists data from Vault at the given path.
	vault list secret/my-app/	Can be used to list keys in a given secret engine.
login	Authenticate locally	
	vault **login** [-method=TYPE, options] [ARGS K=V...]	Authenticates users or machines to Vault using the provided arguments
	vault login -method=userpass username=my-username	-method specifies the auth method,
		use **vault auth help TYPE** to get details
agent	Start a Vault agent	
	vault **agent** [options]	Starts a Vault agent that can perform automatic authentication in
	vault agent -config=/etc/vault/config.hcl	certain environments.
server	Start a Vault server	
	vault **server** [options]	Starts a Vault server that responds to API requests
	vault server -config=/etc/vault/config.hcl	
	vault server -dev -dev-root-token-id="root"	By default,
		Vault will start in a "sealed" state. The Vault cluster must be initialized
		before use, usually by the "vault operator init" command. Each Vault server must
		also be unsealed using the "vault operator unseal" command or the API before the
		server can respond to requests.
status	Print seal and HA status	
	vault **status** [options]	Prints the current state of Vault including whether it is sealed and
		if HA mode is enabled.
unwrap	Unwrap a wrapped secret	
	vault **unwrap** [options] [TOKEN]	Unwraps a wrapped secret from Vault by the given token.
		If no TOKEN given the current authenticated token is used
audit	Interact with audit devices	
	vault **audit disable** [options] **PATH**	disable the audit device at PATH
	vault **audit disable file/**	
	vault **audit list** [options]	List all enabled audit devices
	vault **audit list -detailed**	
	vault **audit enable** [options] **TYPE** [CONFIG K=V...]	enable an audit device of TYPE
	where TYPE = file, syslog, socket	
	vault **audit enable file** file_path=/var/log/audit.log	
	vault **audit enable syslog** tag="vault" facility="AUTH"	
	vault **audit enable socket** address=127.0.0.1:9090 socket_type=tcp	
auth	An auth method is responsible for authenticating users or machines and assigning them policies with which they can access Vault.	
	vault **auth list** [options]	Lists the enabled auth methods
	vault **auth list -detailed**	
	vault **auth enable** [options] **TYPE**	Enables a new auth method of TYPE at -PATH
	where TYPE=approle, alicloud, aws, azure, gcp, cf, github, jwt, kerber	
	kubernetes, oracle, ldap, okta, radius, cert, token, userpa	
	vault **auth enable -path=userpass userpass**	
	vault **auth disable** [options] **PATH**	Disables an existing auth method at the given PATH
	vault **auth disable userpass/**	
	vault **auth help** [options] **TYPE | PATH**	More detailed help about specific auth TYPES and their usage
	vault **auth help userpass**	
	vault **auth tune** [options] **PATH**	Tunes the configuration options for the auth method at the given PATH
	vault **auth tune -default-lease-ttl=72h github/**	
debug	Runs the debug command	
	vault **debug** [options]	Probes a specific Vault server node for a specified period of time, recording
		information about the node, its cluster, and its host environment. The
		information collected is packaged and written to the specified path.
kv	Interact with Vault's Key-Value storage	
	For KEY secret/a/b/foo, foo is a METADATA header followed by	========== Metadata ==========
	zero or more VERSIONED DATA blocks. DATA blocks are key/value pairs.	cas_required false # setable
	PATH a/b/ are directories that only exists due to files (like git).	delete_version_after 0s # setable
	vault kv commands operate on latest VERSIONED DATA block.	max_versions 0 # setable
	vault kv metadata commands operator on the METADATA header.	current_version 3
	Deleting the metadata, deletes the entire key (and all data).	oldest_version 0
		created_time 2020-11-24T03:23:48.044913Z
		updated_time 2020-11-24T20:56:21.807882Z
		====== Version 1 ======
		created_time 2020-11-24T03:23:48.044913Z
		deletion_time n/a
		destroyed false
		data K=V, ..., K=V
		====== Version 2 ======
		created_time 2020-11-24T20:56:18.10632Z
		deletion_time n/a
		destroyed false
		data K=V, ..., K=V

KV CONFIG DEFAULTS	vault read secret/config vault write secret/config cas-required=true delete-version-after=. max-versions=.	To set kv secret engine configuration defaults NOTE: not kv commands
CAS	vault kv metadata put -cas-required secret/foo vault kv put -cas=1 secret/foo bar=baz	Prevent unintentional changes. Once check-and-set is enabled, every write operation requires the cas parameter with the current verion of the secret. Set cas to 0 when a secret at that path does not already exist.
	vault **kv list** [options] **PATH** vault kv list secret/my-app # list all files under my-app	Lists data from Vault's key-value store at the given path.
	vault **kv delete** [options] **KEY** vault kv delete secret/foo # delete latest version of foo vault kv delete -versions=3 secret/foo # delete version 3 of foo	Deletes the data for the provided version and path in the key-value store. The versioned data will not be fully removed, but marked as deleted and will no longer be returned in normal get requests.
	vault **kv undelete** [options] **KEY** vault kv undelete -versions=3 secret/foo	Undeletes the data for the provided version and path in the key-value store.
	vault **kv destroy** [options] **KEY** vault kv destroy -versions=3 secret/foo # destroy version 3 of key fo	Permanently removes versions in the KV store
	vault **kv enable-versioning** [options] **KEY**	Turns on versioning for the backend at the provided path
	vault **kv get** [options] **KEY** vault kv get secret/foo # get latest version of foo vault kv get -version=1 secret/foo # get version 1 of foo vault kv get -field=username secret/foo # get the username field of f	Retrieves data from the KV store
	vault **kv put** [options] **KEY K=V | K=- | @file** vault kv put secret/foo team=operations vault kv put secret/foo @data.json # file contains dictionary vault kv put secret/foo bar=- # value read from stdin	Writes data to a new version of KEY. Existing data is ignored
	vault **kv patch** [options] **KEY K=V | K=- | @file**	Merges data into a new version of KEY. Existing data is merged.
	vault **kv rollback** [options] **KEY** vault kv rollback -version=2 secret/foo # make v2 the latest version	Promote a given version to become the latest version at the given path.
	vault **kv metadata get** [options] **KEY** vault kv metadata get secret/foo # get all versions of foo	Get all metadata about all versions of the key
	vault **kv metadata put** [options] **KEY** -cas-required -delete-version-after=<duration> -max-versions=<int>	create a blank key in the key-value store or update key configuration for a specified key.
	vault **kv metadata delete** [options] **KEY** vault kv metadata delete secret/foo # delete all versions of foo	Permanently deletes all versions and metadata for the provided key.
lease	Interact with leases	
	vault **lease renew** [options] **ID** vault lease renew -increment=120 database/creds/readonly/2f6a614c...	Renews the lease on a secret, extending the time that it can be used before it is revoked by Vault
	vault **lease revoke** [options] **ID** vault lease revoke -prefix aws/creds/deploy	Revoke a lease by ID or prefix
monitor	Stream log messages from a Vault server	
	vault **monitor** [options] vault **monitor -log-level=trace**	stream log messages of a Vault server
namespace	Interact with namespaces	
	vault **namespace list** [options] vault namespace list	list all namespaces
	vault **namespace lookup** [options] **PATH** vault namespace lookup ns1/	looup an existing namespace
	vault **namespace create** [options] **PATH** vault namespace create ns1/	create a namespace
	vault **namespace delete** [options] **PATH** vault namespace delete ns1/	delete a namespace
operator	Perform operator-specific tasks	
	vault **operator init** [options] vault operator init -key-shares=8 -key-threshold=6	Initializes backend for the first time. Shamir's secret sharing algorithm is used to split a newly generated master key into the specified number of key shares such that the specified subset of thos key shares must come together to regenerate the master key. The share are called "unseal keys"
	vault **operator generate-root** [options] [KEY] * see detailed section	Generates a new root token by combining a quorum of share holders.
	vault **operator rekey** [options] [KEY] * see detailed section	Generates a new set of unseal keys. This operation is zero downtime, but it requires the Vault is unsealed and a quorum of existing unseal keys are provided.
	vault **operator migrate** [options] vault operator migrate -config=migrate.hcl	migrate Migrates Vault data between storage backends. Operates directly on encrypted data and does not require a Vault server nor unsealing.
		raft Interact with Vault's raft storage backend
	vault **operator key-status** [options]	Provides information about the active encryption key. Specifically, the current key term and the key installation time.
	vault **operator rotate** [options]	Rotates the underlying **encryption key** which is used to secure data written to the storage backend. This installs a new key in the key ring. This new key is used to encrypted new data, while older keys in the ring are used to decrypt older data. This is an online operation and does not cause downtime.
	vault **operator step-down** [options]	Forces Vault server to step-down from leader to standby
	vault **operator seal** [options]	Seals the Vault server. It will no respond unless unsealed.
	vault **operator unseal** [options]	Unseals the Vault server using Unseal Keys
path-help	Retrieve API help for paths	
	vault **path-help** [options] **PATH** vault path-help database/roles/	Retrieves API help for paths.
plugin	Interact with Vault plugins and catalog	
	vault **plugin deregister** [options] **TYPE NAME** where TYPE = auth, database, secret vault plugin deregister auth my-custom-plugin	Deregister an existing plugin in the catalog

	`vault` **`plugin info`** `[options]` **`TYPE NAME`**
	`vault plugin info database mysql-database-plugin`
		Display information about a plugin in the catalog with the given NAME

	`vault` **`plugin list`** `[options] [TYPE]`	Lists available plugins registered in the catalog

	`vault` **`plugin register`** `[options]` **`TYPE NAME`**
	`vault plugin register -sha256=d3f0a8b... auth my-custom-plugin`
		Register a new plugin in the catalog

	`vault` **`plugin reload`** `[options]`
	`vault plugin reload -plugin=my-custom-plugin`
	`vault plugin reload -mounts=xyz`
		Reload mounted plugin. Either name or mount(s) must be provided, but not both. Specify -scope=global for replicated reloads

	`vault` **`plugin reload-status RELOAD_ID`**
	`vault plugin reload-status d60a...3e83`
		Retrieves the status of a recent **cluster** plugin reload.

policy	Interact with policies	

	`vault` **`policy delete`** `[options]` **`NAME`**
	`vault policy delete my-policy`
		Deletes the policy named NAME in the Vault server.
		Tokens using this policy are affected immediately.

	`vault` **`policy fmt`** `[options]` **`PATH`**
	`vault policy fmt my-policy.hcl`
		Overwrite the file at the given PATH with the properly-formatted policy file contents.

	`vault` **`policy list`** `[options]`	Lists the names of the policies that are installed on the Vault server.

	`vault` **`policy read`** `[options]` **`NAME`**
	`vault policy read my-policy`
		Prints the contents and metadata of the Vault policy named NAME

	`vault` **`policy write`** `[options]` **`NAME PATH`**
	`vault policy write my-policy /tmp/policy.hcl`
	`cat my-policy.hcl | vault policy write my-policy -`
		Uploads a policy with name NAME from the contents of a local file PATH or stdin

print	Prints runtime configurations	
	`vault` **`print token`**	Prints the vault token currenty in use
secrets	Interact with secrets engines	

	`vault` **`secrets disable`** `[options]` **`PATH`**
	`vault secrets disable aws/`
		Disables a secrets engine at the given PATH
		All secrets created by this engine are revoked and its Vault data is removed.

	`vault` **`secrets enable`** `[options,` **`-path=PATH`**`]` **`TYPE`**
	`vault secrets enable -path=amazon aws`
	`vault secrets enable -max-lease-ttl=30m database`
		Enables a secrets engine of TYPE at PATH
		If no PATH is specified, type is used

	`vault` **`secrets list`** `[options]`
	`vault secrets list -detailed`
		Lists the enabled secret engines on the Vault server.
		A TTL of "system" indicates that the system default is in use.

	`vault` **`secrets move`** `[options]` **`SRCPATH DSTPATH`**
	`vault secrets move secret/ generic/`
		Moves an existing secrets engine to a new path.
		Any leases from the old secrets engine are revoked

	`vault` **`secrets tune`** `[options]` **`PATH`**
	`vault secrets tune -default-lease-ttl=72h pki/`
		Tunes the configuration options for the secrets engine at the given PATH

ssh	Initiate an SSH session	

	`vault` **`ssh`** `[options]` **`username@ip`** `[ssh options]`
	`   where -mode=ca, dynamic, otp`
	`vault ssh -mode=otp -role=my-role user@1.2.3.4`
		Establishes an SSH connection with the target machine.

token	Interact with tokens	
TOKEN TYPES	Periodic: Renews for a fixed amount of time indefinitly Use Limited: Expires at the end of their last use Orphan: Has no parent. Expires independantly when TTL, MaxTTL, use count expires	

	`vault` **`token capabilities`** `[options] [TOKEN]` **`PATH`**
	`vault token capabilities 96dd...f4bc secret/foo`
		Print capabilities of TOKEN for a given PATH (as defined by policies)
		If no TOKEN is specified the locally authenticated token is used

	`vault` **`token create`** `[options]`
	`vault token create -ttl=30 -policy=default`
	`vault token create -role=token-role`
	`vault token create -use-limit=2`
	`vault token create -orphan`
		Create child token with all POLICIES & PERMISSIONS of current authenticated token unless a subset of policies is specified.
		Token expires after TTL unless renewed
		TYPE can be service or batch

	`vault` **`token lookup`** `[options]` **`TOKEN | -accessor ACCESSOR`**
	`vault token lookup TOKEN             # does not consume usage`
	`VAULT_TOKEN=TOKEN vault token lookup # consumes a usage`
		Displays information about a TOKEN or ACCESSOR.
		If no TOKEN is specified the locally authenticated token is used

	`vault` **`token renew`** `[options]` **`TOKEN | -accessor ACCESSOR`**
	`vault token renew -increment=30m -accessor ACCESSOR`
		Renews a token's lease, extending the amount of time it can be used.

	`vault` **`token revoke`** `[options]` **`TOKEN | -self | -accessor ACCESSOR`**
	`vault token revoke -mode=orphan`
		MODE unspecified, Revoke token and all of the token's children.
		MODE = orphan, Revoke token only, leaving the children as orphans.
		MODE = path, Revoke tokens and children from a given path prefix

Generate A Root Token

```
Start a root token generation (end with vault operator generate-root -cancel)

vault operator generate-root -init
>>> OTP   WnOHKZq9pC6ElJW6qIQfLmFHAV
>>> NONCE 03bed1c3-f0bb-7a04-2436-0c461ba9bf43

Run for each Unseal Key using the same NONCE

vault operator generate-root -nonce=$NONCE
>>> ENCODED_TOKEN JEAGDQYRNUAFK1NwFB8FWD85GzQcAiIHCmQ

Decode the Encoded Token

vault operator generate-root -otp=$OTP -decode=$ENCODED_TOKEN
>>> s.IEMKDyuhe5xURnNpJRPodOK2
```

Rekey a Vault (generate a new master key and shared keys)

```
Start a rekey with new values for shares and threshold (end with vault operator rekey -cancel)

vault operator rekey -init -key-shares=3 -key-threshold=3

Key                        Value
---                        -----
Nonce                      7e40b8dd-69d6-fa28-40c3-bd6de319a8ff
Started                    true
Rekey Progress             0/1
New Shares                 3
New Threshold              3
Verification Required      false

Run for each Unseal Key using the same NONCE

vault operator rekey -nonce=7e40b8dd-69d6-fa28-40c3-bd6de319a8ff

Key 1: g882yYzwHtNWnAM6uqEpdNkN8G9iga6ax5wmvGChEPC9
Key 2: oKnQf5hPBabE3hZ8QllnBWCVMa05uH2/VM6gUhoTSlah
Key 3: EVtjMBIVOnuaiQt+CoimUtgXAhyegyYncPIo61QSGrh3
```