# Vault Cheat Sheet

**Contents**



## CONFIG

create CONF file /etc/vault/vault.conf

```
backend "file" {
   path = "vault"
}

listener "tcp" {
   address = "0.0.0.0:8200"
   tls_disable = 1
}
```

* path refers to the path on your OS, so here it will be /vault
* bind the listener address to 0.0.0.0 not 127.0.0.1, otherwise you wont be able to authenticate from other machines

```
export VAULT_ADDR=http://127.0.0.1:8200
```

or add to your ENV file /etc/vault/vault_env,

```
export VAULT_ADDR=http://127.0.0.1:8200
export VAULT_UNSEAL_KEY=66JulX5NeL2z0xPjNFqOnCKQh3WHIYCc0nlPYZawRtU=
export VAULT_ROOT_TOKEN=39dafbec-0631-09e3-293e-231dcdd0a3a9

source vault_env
```

---

## START/STOP

start an unsealed DEV test server (you will be authenticated in as root automatically)

```
vault server -dev
```

start a normal server

```
vault server -config /etc/vault/vault.conf &
```

Stop vault server

```
ps -ef | grep "vault server" | grep -v grep | awk '{print $2}' | xargs kill -9
```

or use a Service script

---

## INIT

This initializes Vault with 1 key and 1 threshold (only 1 key is needed to unseal or open the vault). For more secure implementation, use multiple keys and have a threshold of at least 2, meaning you need at least 2 keys provided to open the vault

`cmd`

```
vault init -key-shares=1 -key-threshold=1 -tls-skip-verify
```

curl

```
curl -X PUT -d "{\"secret_shares\":1, \"secret_threshold\":1}" http://127.0.0.1:8200/v1/sys/init
```


```
{
"keys":["8d8e174384b37456198d1803f4a72b6370d855ff9f8f426b48b88c9750b37381"],
"keys_base64":["jY4XQ4SzdFYZjRgD9KcrY3DYVf+fj0JrSLiMl1Czc4E="],
"root_token":"ac36e083-cd31-3f2c-5f0d-d6dd29fb4ae9"
}
```


export ENV
export VAULT_ROOT_TOKEN=ac36e083-cd31-3f2c-5f0d-d6dd29fb4ae9
export VAULT_UNSEAL_KEY=8d8e174384b37456198d1803f4a72b6370d855ff9f8f426b48b88c9750b37381

or add VAULT_ROOT_TOKEN, VAULT_UNSEAL_KEY to your ~/.bashrc

### check Status

cmd

```
vault status
```

**RE-INIT UNSEAL KEYS**

If you need to re-init vault (and generate new unseal keys), delete the local backend storage, so if your vault.conf is
path = "vault"

run
`rm -rf /vault`

then start, stop vault service, run init again

---

## UNSEAL

Before adding any passwords, unseal the vault using the Unseal Key from above step. Unsealing makes Vault available for operations, it should only be sealed in event of a breach.

### unseal Vault

cmd

```
vault unseal $VAULT_UNSEAL_KEY
```

curl

```
curl -X PUT -d '{"key": "8d8e174384b37456198d1803f4a72b6370d855ff9f8f426b48b88c9750b37381"}'
http://127.0.0.1:8200/v1/sys/unseal
```


```
{"sealed":false,"t":1,"n":1,"progress":0,"nonce":"","version":"0.6.5","cluster_name":"vault-cluster-
b47dfa63","cluster_id":"abbb17c0-faad-e0b8-8dc1-8bd2db93e39b"}
```


### check seal/unseal status

curl

```
curl -X GET -H "X-Vault-Token:$VAULT_ROOT_TOKEN" http://127.0.0.1:8200/v1/sys/seal-status
```

**Seal the vault** (will remove Master key)

cmd

curl

```
curl -X PUT -H "X-Vault-Token:$VAULT_ROOT_TOKEN" http://127.0.0.1:8200/v1/sys/seal
```

---

## AUTHENTICATE

if using cmd line, you need to authenticate. If using CURL, dont need to authenticate, just pass your auth token

### Auth into Vault

cmd

```
vault auth $VAULT_ROOT_TOKEN
```

curl

```
automatically provided as -H "X-Vault-Token:$VAULT_ROOT_TOKEN"
```

---

## SECRETS

Backend or a Mount is a file system that Vault uses to store information. Secrets is a generic backend.

### Write a secret

cmd

```
vault write secret/users password=a341xr09
```

curl

```
curl -X POST -H "X-Vault-Token:$VAULT_TOKEN" -d '{"password":"a341xr09"}' http://127.0.0.1:8200/v1/secret/users
```

write multiple values

```
vault write secret/users name=joe lastname=smith age=39
```

read secret

```
vault read secret/users
```

read secret in JSON, use 'jq' to parse JSON output

```
vault read -format=json secret/users | jq .data.password
```

read secret in JSON, use python to parse JSON output

```
vault read -format=json secret/users | python -c 'import sys,json; print json.load(sys.stdin)["data"]["password"]'
```

show all secret keys

```
vault list secret
```

delete secret

vault delete secret/users

if this doesnt work, delete from OS path (if backend=File)

rm -rf /vault/logical/GUID

---

## CUSTOM BACKEND & MOUNTS

custom backends can be created or 'mounted', using "Generic" type

```
vault mount -path myStuff -description="my secrets" generic
```

check mounts

```
vault mounts
```

```
Path        Type     Default TTL   Max TTL    Description
cubbyhole/  cubbyhole n/a           n/a        per-token private secret storage
myStuff/    generic   system        system     my stuff
secret/     generic   system        system     generic secret storage
sys/        system    n/a            n/a       system endpoints used for control
```

write to your custom backend

```
vault write myStuff/info id=123 region=US rank=3
```

```
vault read myStuff/info
Key Value
--- -----
```

`refresh_interval 768h0m0s`
`id 123`
`rank 3`
`region US`

unmount your backend
`vault unmount myStuff`

---

## SECRET BACKEND - CUBBYHOLE

background on cubbyhole
https://www.hashicorp.com/blog/cubbyhole-authentication-principles/

---

## ACL POLICY

**(Access Control List)**
Access control policies in Vault control what a user can access, these are the ultimate controllers of who can see what

for example

```
path "secret/jira/password" {
   policy = "read"
}
```

only allows a read on the password, to whoever is accessing it

create new file called **dev.hcl**

```
name = "dev"

path "secret/*" {
    policy = "write"
}

path "myCorp/projectA/database/password" {
    policy = "read"
}

path "auth/token/lookup-self" {
    policy = "read"
}
```

### write the policy

`cmd`

```
vault policy-write mypolicy ACL.hcl
```

you policy is now written in-memory

### see all written policies

`cmd`

```
vault policies
```

`curl`

```
curl -X GET -H "X-Vault-Token:$VAULT_ROOT_TOKEN"  http://127.0.0.1:8200/v1/sys/policy
```

{"keys":["master","default","acl","root"],"policies":[**"mypolicy**","default","acl","root"],"request_id":"d557373c-962c-e86b-3089-d7671c03c54f","lease_id":"","renewable":false,"lease_duration":0,"data":{"keys":["mypolicy","default","acl","root"],"policies":["mypolicy","acl","root"]},"wrap_info":null,"warnings":null,"auth":null}

### see your specific policy

cmd

```
vault policies mypolicy
```

```
path "secret/*" {
   policy = "write"
}
path "secret/projectA/database/password" {
   policy = "read"
}
path "auth/token/lookup-self" {
   policy = "read"
}
```

---

## AUTH BACKEND - TOKENS

create a token

```
vault token-create
```

```
Key Value
--- -----
token e032a2fd-8c25-1746-f5b6-ef7497d5ed65
token_accessor 7ec939a8-ae11-4ebe-5bba-facf97066167
token_duration 0s
token_renewable false
token_policies [root]
```

create token for specific policy

```
vault token-create -policy=myPolicy
```

revoke a token

```
vault token-revoke
```

authenticate with token (only for cmd line)

```
vault auth 0e2b4e8e-e15d-c2b0-1354-2546ce42fde7
```

revoke all tokens for a secret

```
vault revoke -prefix secret/users/password
```

lookup current token info

```
vault token-lookup
```

generate a new ROOT token (root tokens never expire and have access to everything)

1. unseal Vault
2. generate 1 time password

   ```
   vault generate-root -genotp
   OTP: qIoKVrKsaLOzBqYTxX1r0A==
   ```

3. get encoded root token

   ```
   vault generate-root -otp qIoKVrKsaLOzBqYTxX1r0A==
   2017/03/16 13:43:20.166090 [INFO ] core: root generation initialized: nonce=bff2360c-9366-2385-dc15-fc842a0a83a5
   Root generation operation nonce: bff2360c-9366-2385-dc15-fc842a0a83a5

   Key(will be hidden): provide $VAULT_UNSEAL_KEY here

   2017/03/16 13:51:13.114477 [INFO ] core: root generation finished: nonce=bff2360c-9366-2385-dc15-fc842a0a83a5
   Nonce: bff2360c-9366-2385-dc15-fc842a0a83a5
   Started: true
   Rekey Progress: 1
   Required Keys: 1
   Complete: true
   ```

```
Encoded root token: JilLZtsUVHzwUHU2rMMcvg==
```

4. decode encoded root token

```
vault generate-root -otp qIoKVrKsaLOzBqYTxX1r0A== -decode=JilLZtsUVHzwUHU2rMMcvg==
Root token: 8ea34130-69b8-3ccf-4356-d32569be776e
```

---

## AUTH BACKEND - AppRoles

### check available auth methods

cmd

```
vault auth -methods
```

### enable approle

cmd

```
vault auth-enable approle
```

curl

```
curl -X POST -H "X-Vault-Token:$ROOT_VAULT_TOKEN" -d '{"type":"approle"}' http://127.0.0.1:8200/v1/sys/auth/approle
```

### create AppRole

cmd

```
vault write -f auth/approle/role/nyc-admins
```

curl

```
curl -X POST -H "X-Vault-Token:$ROOT_VAULT_TOKEN" -d '{"policies":"dev-policy,test-policy"}'
http://127.0.0.1:8200/v1/auth/approle/role/testrole
```

### get Role ID

cmd

```
vault read auth/approle/role/testrole/role-id
```

curl

```
curl -X GET -H "X-Vault-Token:$ROOT_VAULT_TOKEN" http://127.0.0.1:8200/v1/auth/approle/role/testrole/role-id | jq .
```

### get Secret ID for role

cmd

```
vault write -f auth/approle/role/testrole/secret-id
```

curl

```
curl -X POST -H "X-Vault-Token:$VAULT_TOKEN" http://127.0.0.1:8200/v1/auth/approle/role/testrole/secret-id | jq .
get Token via Role
```

### login with Role

cmd

```
vault write auth/approle/login role_id=ROLE_ID secret_id=SECRET_ID
```

curl

```
curl -X POST \ -d '{"role_id":"988a9dfd-ea69-4a53-6cb6-9d6b86474bba","secret_id":"37b74931-c4cd-d49a-9246-
ccc62d682a25"}' \ http://127.0.0.1:8200/v1/auth/approle/login | jq .
```

### EXAMPLE

**get MYSQL passwords making calls from another machine**

create policy 'mysql'

mysql.hcl

```
path "sys/*" {
   policy = "deny"
}

path "my_corp/mysql/*" {
   policy = "read"
}
```

create role called 'nyc-admins'

```
vault write -f auth/approle/role/nyc-admins
```

associate Role to a set of policies
```
vault write auth/approle/role/nyc-admins policies=mysql, devs
```

get the Role ID of the role
```
vault read auth/approle/role/nyc-admins/role-id

Key     Value
---     -----
role_id ca1dbec4-37f1-61a2-8a83-87a3d980d8b9
```

get a Secret ID for the role
```
vault write -f auth/approle/role/nyc-admins/secret-id

Key                 Value
---                 -----
secret_id           445f6eab-4207-a45b-b6b8-a3e86f128fcc
secret_id_accessor c7da2183-3d68-31c6-70ef-b0d9081e6ceb
```

get a token cred for this role
```
vault write auth/approle/login role_id=ca1dbec4-37f1-61a2-8a83-87a3d980d8b9 secret_id=445f6eab-4207-a45b-b6b8-
a3e86f128fcc
```

save it as $VAULT_TOKEN

from machine123, get the credentials for mysql

```
curl -X GET -H "X-Vault-Token:$VAULT_TOKEN" http://<IP of Vault Server>:8200/v1/secrets/mysql

{"request_id":"18b7ed7b-d349-6132-3ea4-
20e4dbd6d9a5","lease_id":"","renewable":false,"lease_duration":2764800,"data":
{"pw":"abcdef","server":"mysql23.corp"},"wrap_info":null,"warnings":null,"auth":null}
```