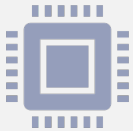
HASHICORP

Analisi condotta
da Luisa Mele

VAULT



È un Secret Management Tool, ovvero la gestione di informazioni che utilizziamo per autorizzare o fare il login a determinate informazioni ‘’sensibili’’. Facendo un esempio potrebbero essere:

Username e password;

API tokens;

DOVE SONO STORATE LE CHIAVI?

Le chiavi sono storate:

01

Localmente nelle macchine degli utenti (Azure or AWS keys);

02

Password files condivise tra team members;

03

List of Confluence keys in plaintext e plaintext in repositories (è pericoloso) ;

04

Environment variables (più sicuro);

HASICORP CHARACTERISTIC

- General secret storage;
- API key generation for scripts;
- Employee Generation Storage;
- Data Encryption;

Il Vault opererà come un client-server application dove il client interagirà con il back-end tramite TLS encryption connection.

La key non può essere utilizzata senza il Vault Server, quindi nessuno può avere accesso a queste key, se non le persone autorizzate.

RETREIVE THE SECRET

- Per recuperare la chiave possiamo utilizzare la CLI:

```
version                 1
christophe@Christophes-MacBook-Pro ~ % vault kv get -mount=secret hello
```

```
christophe@Christophes-MacBook-Pro ~ % vault kv get -mount=secret hello
== Secret Path ==
secret/data/hello

======= Metadata =======
Key                Value
---                -----
created_time       2023-02-24T19:10:03.061627Z
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            1

=== Data ===
Key    Value
---    -----
foo    world
```

ABILITARE E DISABILITARE IL VAULT

- Possiamo, tramite CLI abilitare e disabilitare i secret, considerando l'utilizzo che dobbiamo fare del Vault,anche creando dynamic secrets.

```
christophe@Christophes-MacBook-Pro ~ % vault secrets enable -path=aws aws
Success! Enabled the aws secrets engine at: aws/
christophe@Christophes-MacBook-Pro ~ % vault secrets list
Path          Type         Accessor              Description
----          ----         --------              -----------
aws/          aws          aws_bf35d33e          n/a
cubbyhole/    cubbyhole    cubbyhole_489b24b8    per-token private secret storage
identity/     identity     identity_f8ca662d     identity store
secret/       kv           kv_469ec791           key/value secret storage
sys/          system       system_cee4e823       system endpoints used for control
christophe@Christophes-MacBook-Pro ~ %
```

DYNAMIC SECRET

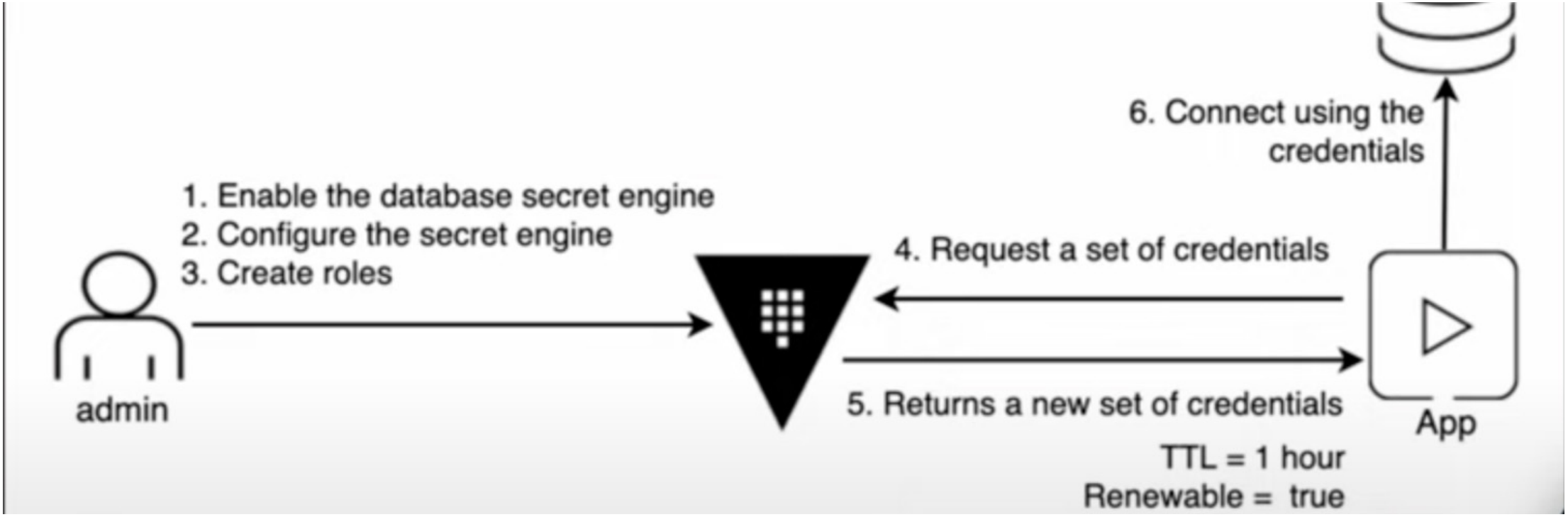
- I secrets sono distrutti una volta utilizzati. Questo utilizzo può essere considerato per Cloud Bases access o DB access.
- Solitamente gli utenti generano un set statico di DB credentials, che poi vengono storate come environment variables o in plaintext in configuration files.
- Il Vault risolve il problema.

```
export USER="username"
export PASSWORD="password"
```

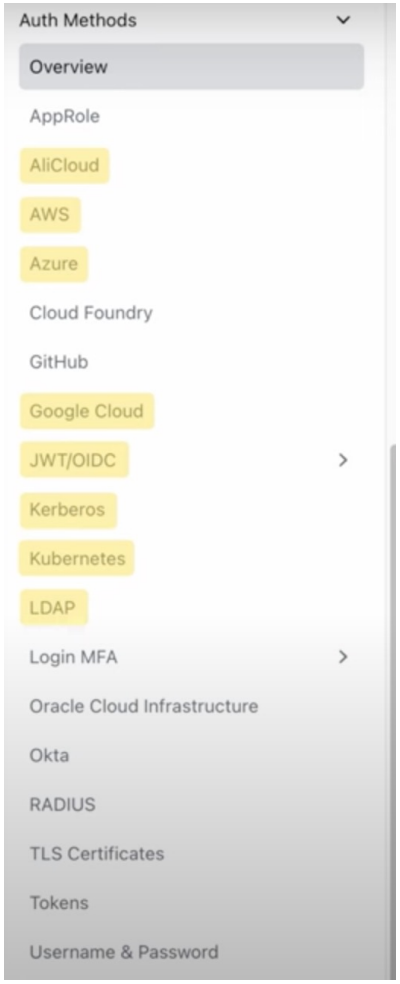
```
from dotenv import load_dotenv   #for python-dotenv method
load_dotenv()                    #for python-dotenv method

import os

user_name = os.environ.get('USER')
password = os.environ.get('password')

print(user_name, password)

# output

username password
```

COME RISOLVERE?

- Il DB chiede le credenziali al Vault e se si è autorizzati le fornisce, in caso contrario inibisce l'accesso alle risorse.
- Il Vault ha un TTL (Time to leave), nel range del quale le credenziali sono valide; dopo di che il Vault genererà nuove credenziali.

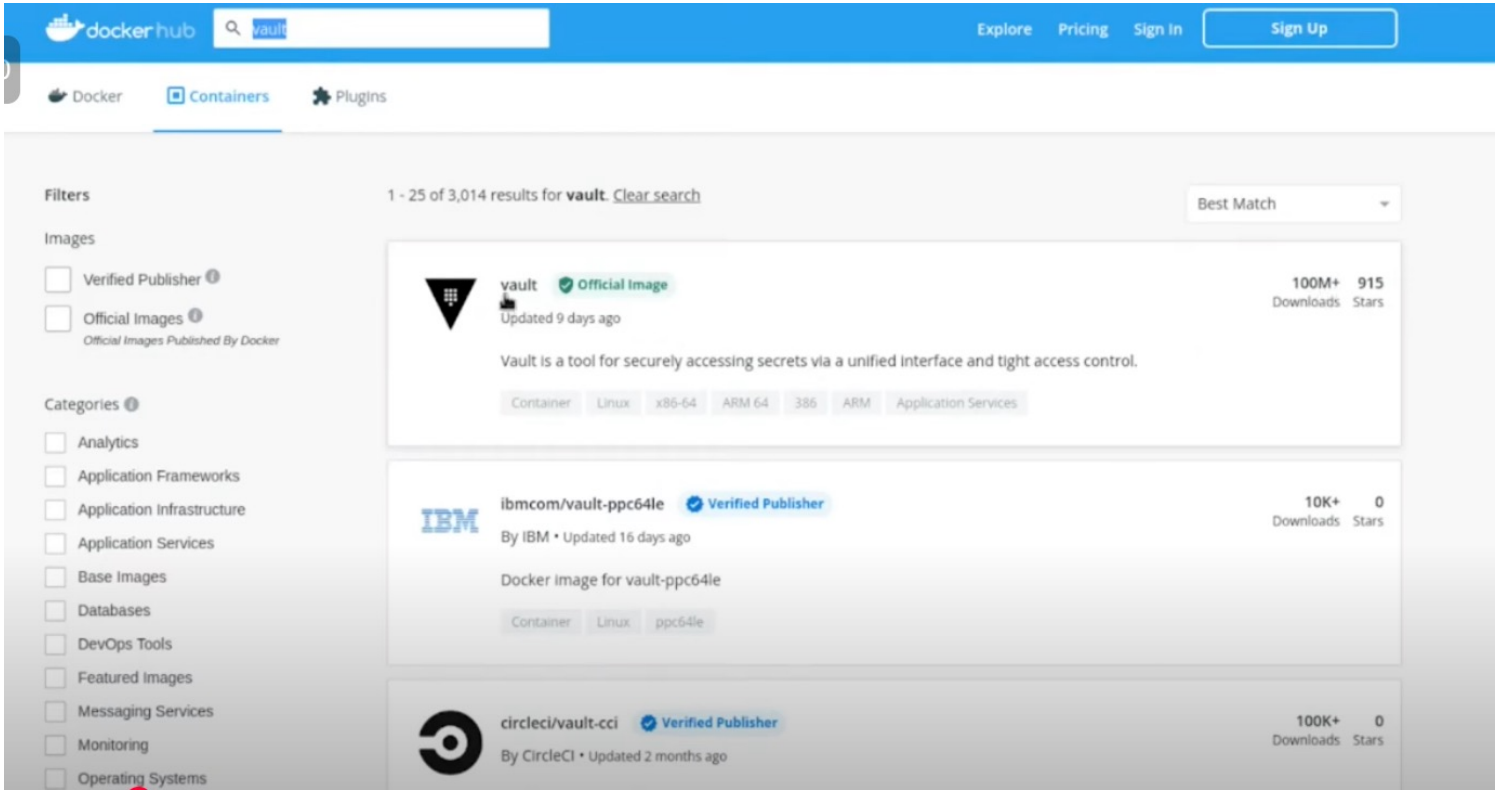


AUTH METHODS



VAULT PUÒ ESSERE UTILIZZATO VIA:

- CLI;
- API CALL;
- INTERFACE.



VAULT CONNECTION

Il vault può essere connesso ad esempio a Docker, scaricando la Docker Vault Official image

DOCKER INSTALLATION VAULT

- Da CLI possiamo digitare ‘’docker pull vault’’
- Per mandare in esecuzione il vault su Docker:

```
$ docker run --cap-add=IPC_LOCK -e 'VAULT_DEV_ROOT_TOKEN_ID=myroot' -e 'VAULT_DEV_LISTEN_ADDRESS=0.0.0.0:1234' vault
```

```
[tobias@localhost ~]$ docker run -d --rm --name vault-server --cap-add=IPC_LOCK -e
'VAULT_DEV_ROOT_TOKEN_ID=tdc-token' -e 'VAULT_DEV_LISTEN_ADDRESS=0.0.0.0:1234' vaul
t
```

```
[tobias@localhost ~]$ docker run -d --rm --name vault-server --cap-add=IPC_LOCK -e
'VAULT_DEV_ROOT_TOKEN_ID=tdc-token' -e 'VAULT_DEV_LISTEN_ADDRESS=0.0.0.0:8200' vaul
t
895701b73e3b67f45b48cff6daf5ee7f96379079140b385a78db11189261a540
[tobias@localhost ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PO
RTS       NAMES
895701b73e3b   vault     "docker-entrypoint.s…"   9 seconds ago   Up 7 seconds   82
00/tcp    vault-server
[tobias@localhost ~]$
```

PER RECUPERARE L'IP ADDRESS DI DOCKER

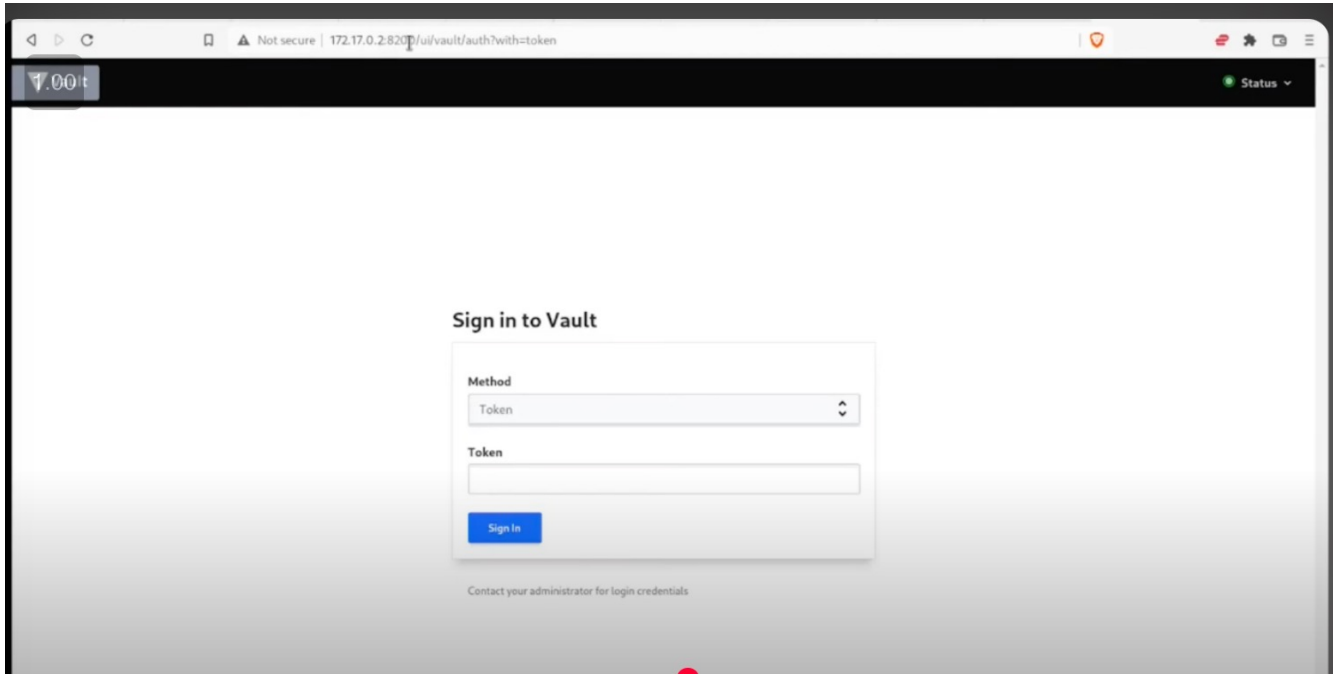
Tramite CLI per connetterci a docker abbiamo bisogno dell'ip address

```
[tobias@localhost ~]$ docker inspect vault-server | grep IPAddress
```

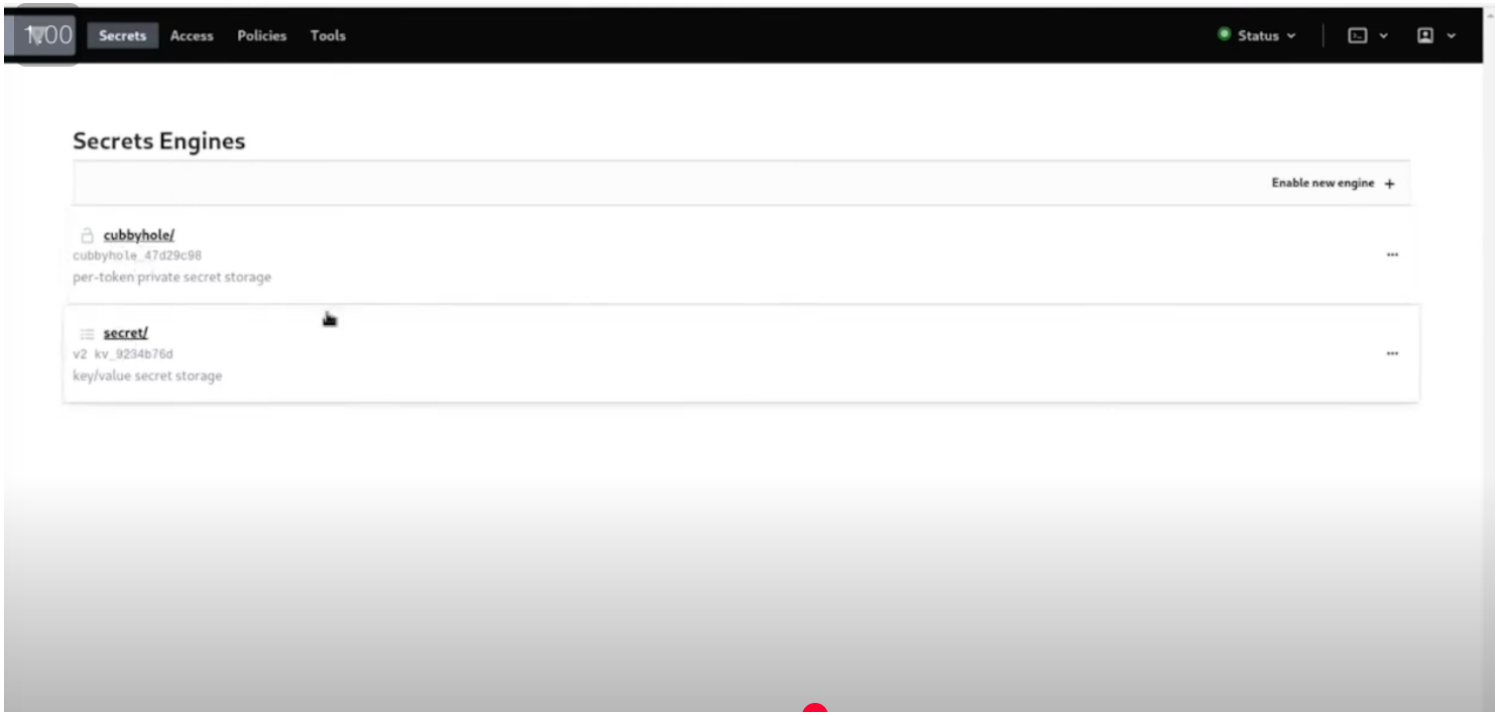
Facciamo l'export in environment var

```
[tobias@localhost ~]$ docker inspect vault-server | grep IPAddress
        "SecondaryIPAddresses": null,
        "IPAddress": "172.17.0.2",
                "IPAddress": "172.17.0.2",
[tobias@localhost ~]$ export VAULT_ADDR='http://172.17.0.2:8200'
[tobias@localhost ~]$
```

COSA VEDIAMO NELL'INTERFACE?



Se abbiamo seguito tutti i passaggi correttamente, nel web browser, tramite interfaccia dovremmo vedere l' Ip address nella barra di ricerca



INTERFACCIA E SECRETS

Nell'interfaccia dovremmo visualizzare tutti i secret engine e creare nuove engine

CONNESSIONE AL VALUT

- Per connetterci al vault:

```
[tobias@localhost ~]$ vault login
Token (will be hidden):
```

- Ci chiederà, poi, il token e se siamo riusciti ad

Accedere visualizzeremo:

```
[tobias@localhost ~]$ vault login
Token (will be hidden):
Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.

Key                  Value
---                  -----
token                tdc-token
token_accessor       zj3UqHdEJhR22s2IGTTuDQNF
token_duration       ∞
token_renewable      false
token_policies       ["root"]
identity_policies    []
policies             ["root"]
[tobias@localhost ~]$
```

CREARE IL SECRET

- Per creare il nostro secret:

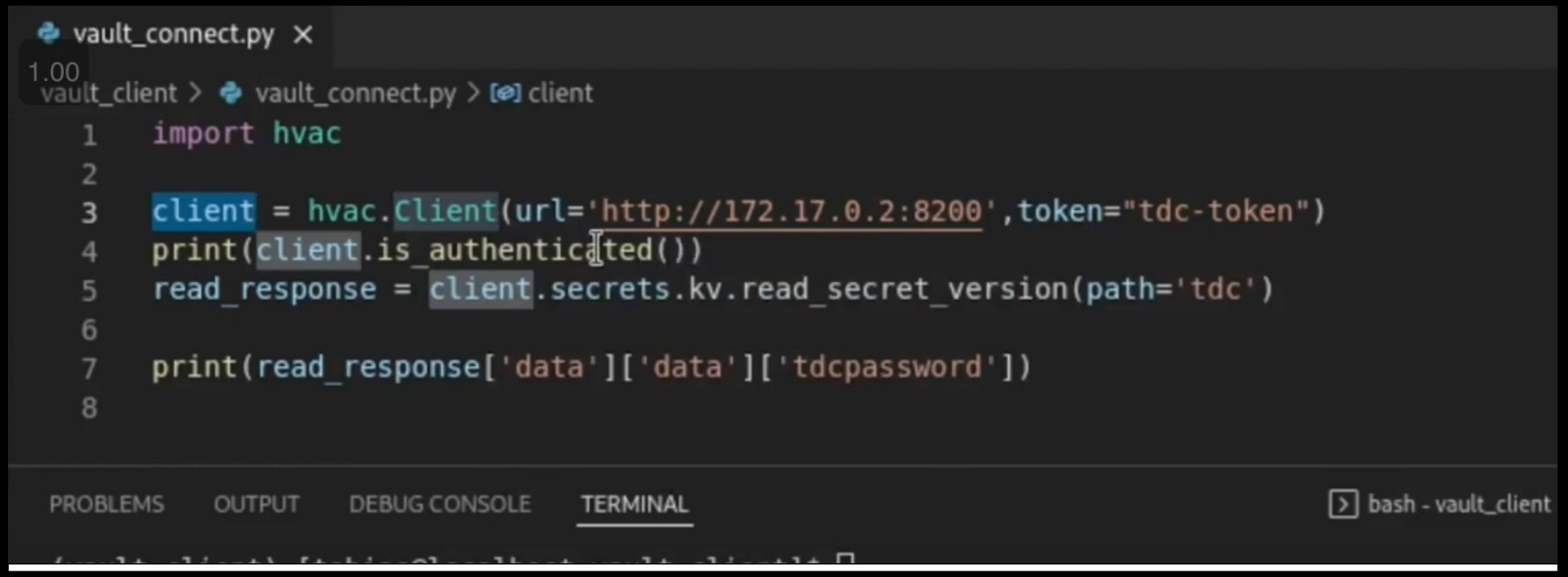
```
[tobias@localhost ~]$ vault kv put secret/tdc tdcpassword=test1234
```

```
[tobias@localhost ~]$ vault kv put secret/tdc tdcpassword=test1234
Key                 Value
---                 -----
created_time        2022-02-06T18:37:45.202247945Z
custom_metadata     <nil>
deletion_time       n/a
destroyed           false
version             1
[tobias@localhost ~]$
```

RECUPERARE IL SECRET

Per recuperare il secret:

```
[tobias@localhost ~]$ vault kv get secret/tdc
======= Metadata =======
Key                Value
---                -----
created_time       2022-02-06T18:37:45.202247945Z
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            1

======= Data =======
Key            Value
---            -----
tdcpassword    test1234
[tobias@localhost ~]$
```



PYTHON

Da IDE, utilizziamo Python per impostare le modalità di recupero e la verifica della connessione tramite hvac