FINDR VAULT DOCUMENTATION

1) Accessing the Vault UI

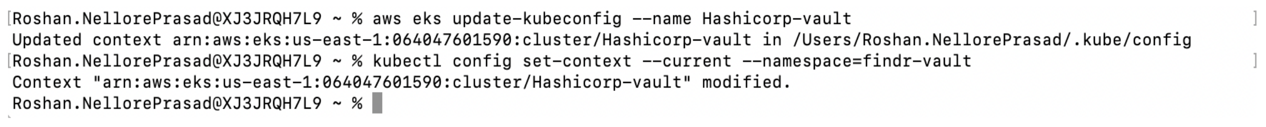
- AUTOMATED PROCESS
  - AWS Configuration(**aws-5g.dp.mss.data-science.dev)**
  - Download vault_setup.sh file.
  - **run->** ./vault_setup.sh

- MANUAL PROCESS:

a) Configure the AWS Account in terminal (**aws-5g.dp.mss.data-science.dev)**

b) Set the path to where the Vault is deployed. In our case we are using the Hashicorp Vault cluster where the vault is deployed in Findr-vault namespace

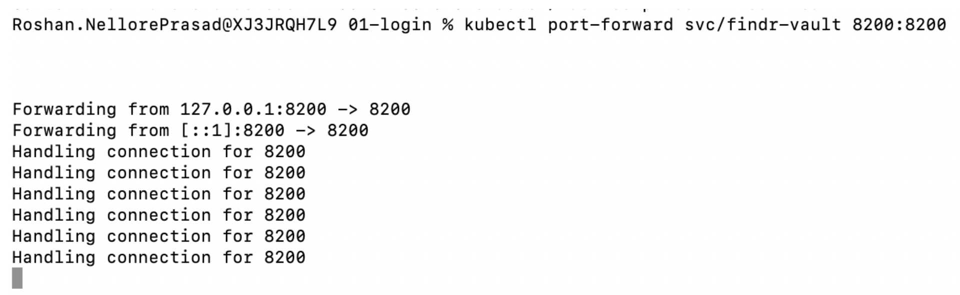
- **aws eks update-kubeconfig --name Hashicorp-vault**
- **kubectl config set-context --current --namespace=findr-vault**



```
Roshan.NellorePrasad@XJ3JRQH7L9 ~ % aws eks update-kubeconfig --name Hashicorp-vault
Updated context arn:aws:eks:us-east-1:064047601590:cluster/Hashicorp-vault in /Users/Roshan.NellorePrasad/.kube/config
Roshan.NellorePrasad@XJ3JRQH7L9 ~ % kubectl config set-context --current --namespace=findr-vault
Context "arn:aws:eks:us-east-1:064047601590:cluster/Hashicorp-vault" modified.
Roshan.NellorePrasad@XJ3JRQH7L9 ~ %
```

c) Port Forward to the service where vault is deployed

- **kubectl port-forward svc/findr-vault 8200:8200**

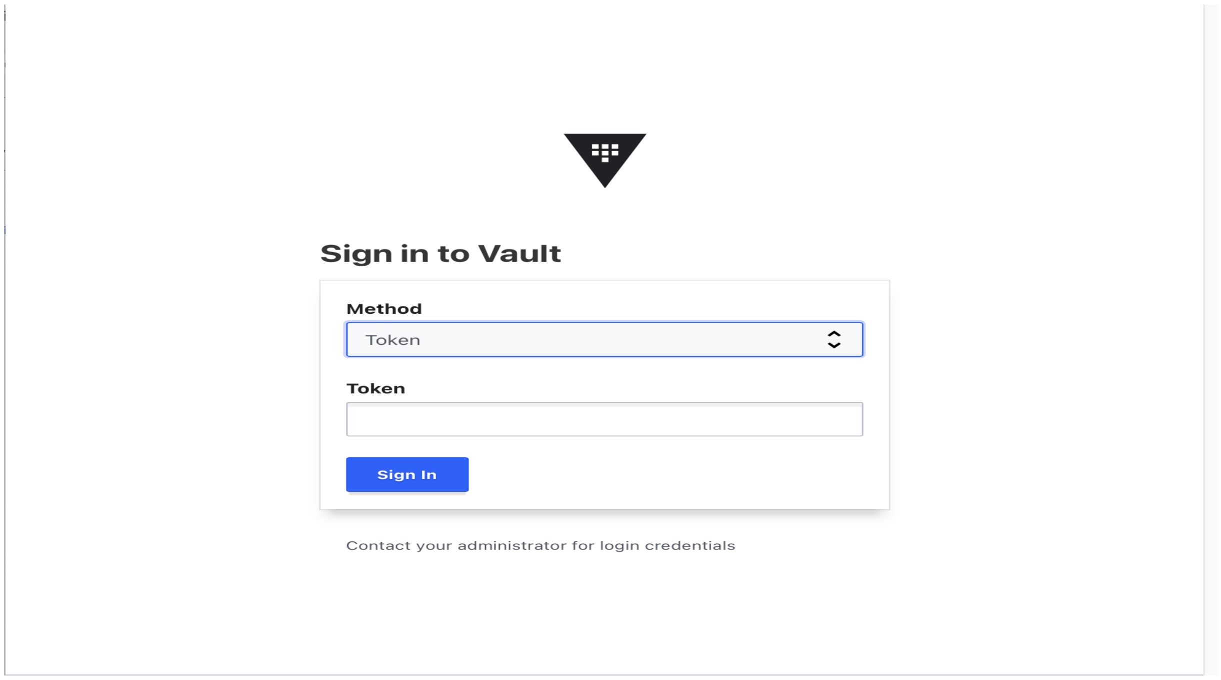


```
Roshan.NellorePrasad@XJ3JRQH7L9 01-login % kubectl port-forward svc/findr-vault 8200:8200


Forwarding from 127.0.0.1:8200 -> 8200
Forwarding from [::1]:8200 -> 8200
Handling connection for 8200
Handling connection for 8200
Handling connection for 8200
Handling connection for 8200
Handling connection for 8200
Handling connection for 8200
```

d) Access the UI

- http://127.0.0.1:8200



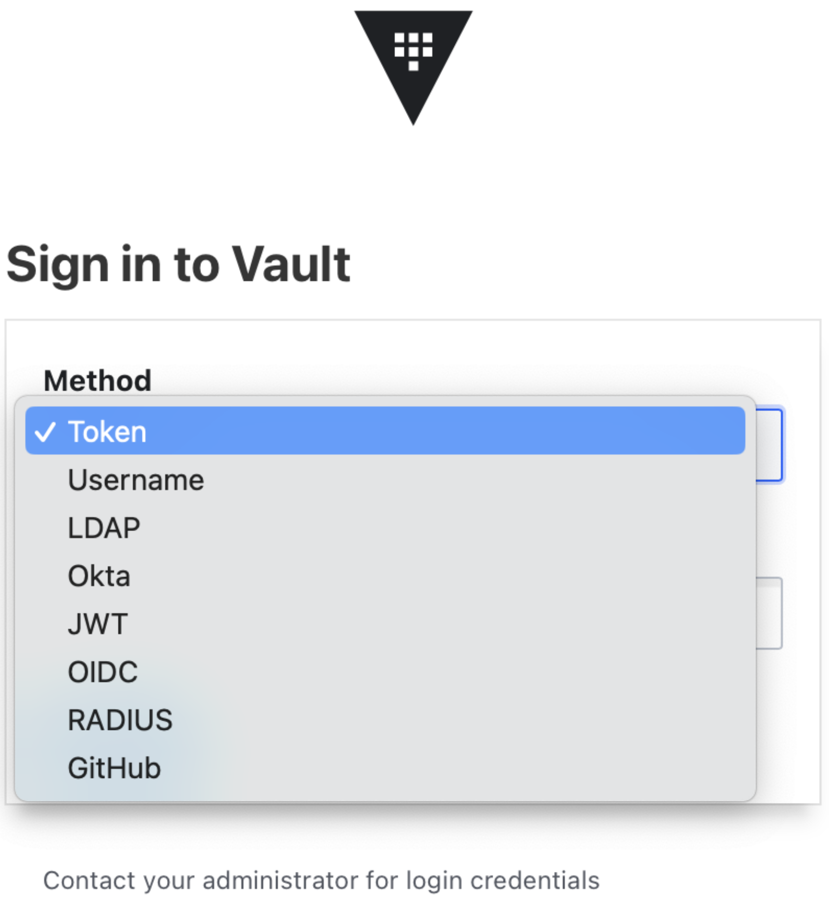
Sign in to Vault

Method: Token

Token:

Contact your administrator for login credentials

e) Authentication methods

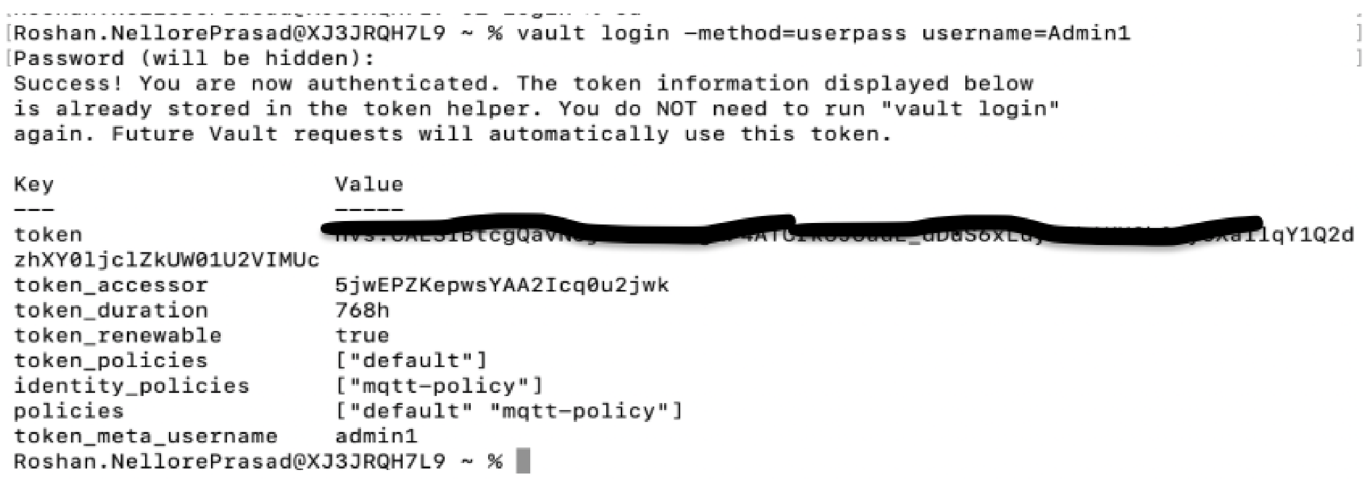


- With **root token**, you could access to every secret, policies, groups, aliases in the Vault
- If you are using other authentication methods, the secrets could be created and viewed based on the policies.

2) VAULT OPERATIONS FROM CLI

a) **USER AUTHENTICATION:**

- Login using AWS credentials
- Use the authentication method you have. Lets see how to access using user authentication method. **[username: Admin1],** similarly there are other authentication methods like token auth, github auth etc..,
    - **vault login -method=userpass username=Admin1**

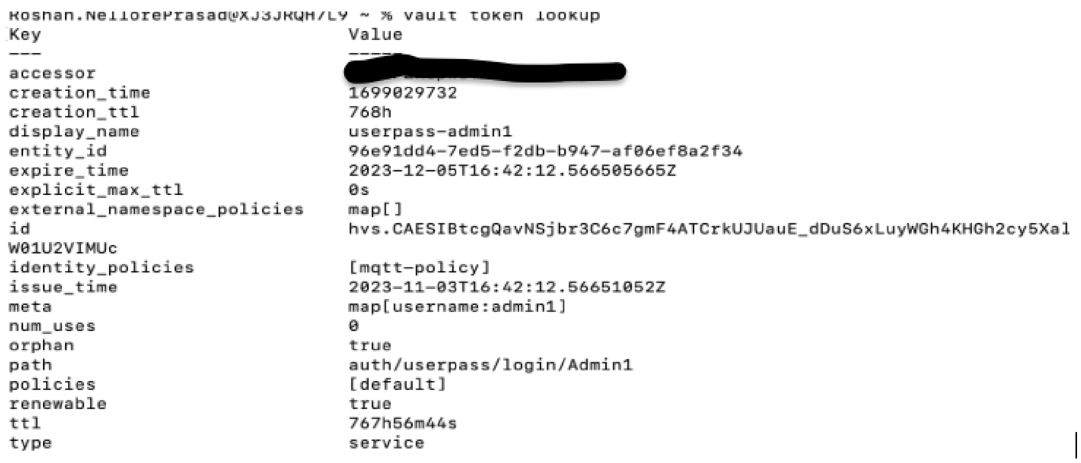


```
Roshan.NellorePrasad@XJ3JRQH7L9 ~ % vault login -method=userpass username=Admin1
Password (will be hidden):
Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.

Key                     Value
---                     -----
token                   [redacted]lqY1Q2d
zhXY0ljclZkUW01U2VIMUc
token_accessor          5jwEPZKepwsYAA2Icq0u2jwk
token_duration          768h
token_renewable         true
token_policies          ["default"]
identity_policies       ["mqtt-policy"]
policies                ["default" "mqtt-policy"]
token_meta_username     admin1
Roshan.NellorePrasad@XJ3JRQH7L9 ~ %
```

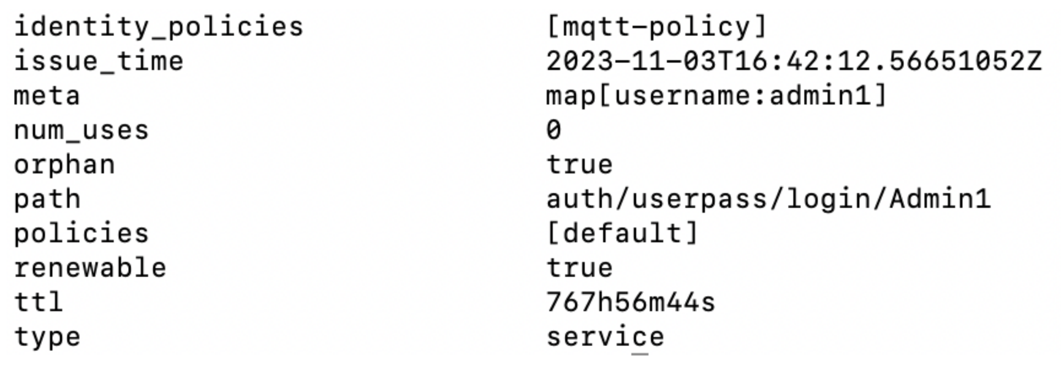
b) **VAULT TOKEN LOOKUP:**

- The token lookup displays information about a token or accessor.
    - **Vault token lookup**



```
Roshan.NellorePrasad@XJ3JRQH7L9 ~ % vault token lookup
Key                            Value
---                            -----
accessor                       [redacted]
creation_time                  1699029732
creation_ttl                   768h
display_name                   userpass-admin1
entity_id                      96e91dd4-7ed5-f2db-b947-af06ef8a2f34
expire_time                    2023-12-05T16:42:12.566505665Z
explicit_max_ttl               0s
external_namespace_policies    map[]
id                             hvs.CAESIBtcgQavNSjbr3C6c7gmF4ATCrkUJUauE_dDuS6xLuyWGh4KHGh2cy5Xal
W01U2VIMUc
identity_policies              [mqtt-policy]
issue_time                     2023-11-03T16:42:12.56651052Z
meta                           map[username:admin1]
num_uses                       0
orphan                         true
path                           auth/userpass/login/Admin1
policies                       [default]
renewable                      true
ttl                            767h56m44s
type                           service
```

c) VAULT IDENTITY POLICY:



```
identity_policies                [mqtt-policy]
issue_time                       2023-11-03T16:42:12.56651052Z
meta                             map[username:admin1]
num_uses                         0
orphan                           true
path                             auth/userpass/login/Admin1
policies                         [default]
renewable                        true
ttl                              767h56m44s
type                             service
```

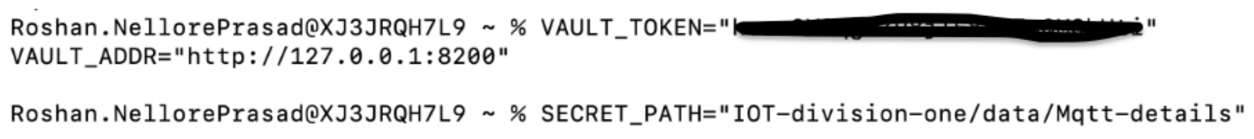
The **vault lookup token** command gives you the output of what you could access in the vault at **identity_policies.** We could see **mqtt-policy** is the secret associated with the Admin1 account.

3) VAULT API's

We could use curl to make api calls. To make an api call, we need to have these 3 details set as environment variables.

1) Vault token
2) Vault address
3) Vault secret path

These details are only visible to root account, who shares it to other users.



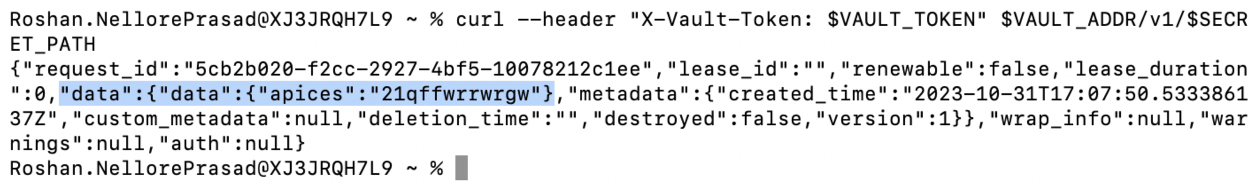
```
Roshan.NellorePrasad@XJ3JRQH7L9 ~ % VAULT_TOKEN="[redacted]"
VAULT_ADDR="http://127.0.0.1:8200"

Roshan.NellorePrasad@XJ3JRQH7L9 ~ % SECRET_PATH="IOT-division-one/data/Mqtt-details"
```

a) API call to read secret:

- curl --header "X-Vault-Token: $VAULT_TOKEN" $VAULT_ADDR/v1/$SECRET_PATH

Once the environment variables are set, we could go ahead and execute the api. The secret **apices** and its password:**21qffwrrwrgw** are visible in terminal.

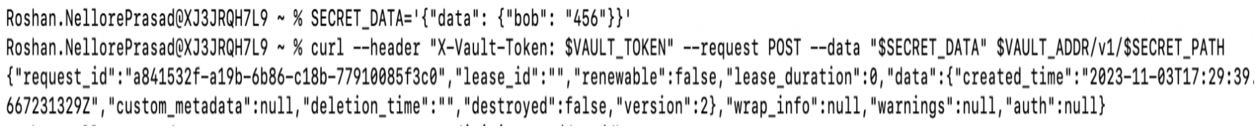


```
Roshan.NellorePrasad@XJ3JRQH7L9 ~ % curl --header "X-Vault-Token: $VAULT_TOKEN" $VAULT_ADDR/v1/$SECR
ET_PATH
{"request_id":"5cb2b020-f2cc-2927-4bf5-10078212c1ee","lease_id":"","renewable":false,"lease_duration
":0,"data":{"data":{"apices":"21qffwrrwrgw"},"metadata":{"created_time":"2023-10-31T17:07:50.5333861
37Z","custom_metadata":null,"deletion_time":"","destroyed":false,"version":1}},"wrap_info":null,"war
nings":null,"auth":null}
Roshan.NellorePrasad@XJ3JRQH7L9 ~ %
```

b) API call to write secret:

In the similar way, you could use API command to write new secrets by adding the environment variable **SECRET_DATA**

- curl --header "X-Vault-Token: $VAULT_TOKEN" --request POST --data "$SECRET_DATA" $VAULT_ADDR/v1/$SECRET_PATH

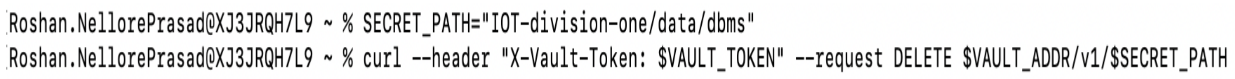


```
Roshan.NellorePrasad@XJ3JRQH7L9 ~ % SECRET_DATA='{"data": {"bob": "456"}}'
Roshan.NellorePrasad@XJ3JRQH7L9 ~ % curl --header "X-Vault-Token: $VAULT_TOKEN" --request POST --data "$SECRET_DATA" $VAULT_ADDR/v1/$SECRET_PATH
{"request_id":"a841532f-a19b-6b86-c18b-77910085f3c0","lease_id":"","renewable":false,"lease_duration":0,"data":{"created_time":"2023-11-03T17:29:39.
667231329Z","custom_metadata":null,"deletion_time":"","destroyed":false,"version":2},"wrap_info":null,"warnings":null,"auth":null}
```

c) API call to delete:

The same way , set the environment variable of the path you wanted to delete.

- $VAULT_TOKEN" --request DELETE $VAULT_ADDR/v1/$SECRET_PATH



```
Roshan.NellorePrasad@XJ3JRQH7L9 ~ % SECRET_PATH="IOT-division-one/data/dbms"
Roshan.NellorePrasad@XJ3JRQH7L9 ~ % curl --header "X-Vault-Token: $VAULT_TOKEN" --request DELETE $VAULT_ADDR/v1/$SECRET_PATH
```