Technical: AWS Secrets Engine

Pre-steps for Vault Installation and Setup

A. Installed vault on mac using homebrew. Reference: https://learn.hashicorp.com/tutorials/vault/getting-started-install
B. Got the error while enabling aws:
   a. “Error enabling: Post "https://127.0.0.1:8200/v1/sys/mounts/aws": dial tcp 127.0.0.1:8200: connect: connection refused” while executing `vault secrets enable aws`
   b. Error indicates CLI is unable to contact your remote Vault server
C. Checked the status of the vault by executing `vault status` and got the same error.
D. Realised vault server was not running. So, I executed `vault server -h`
E. Executed the vault server in “dev” mode referring https://www.vaultproject.io/docs/concepts/dev-server

```
vault server -dev
```

Response:

```
Vault server configuration:

             Api Address: http://127.0.0.1:8200
                     Cgo: disabled
         Cluster Address: https://127.0.0.1:8201
              Go Version: go1.16.5
              Listener 1: tcp (addr: "127.0.0.1:8200", cluster address:
"127.0.0.1:8201", max_request_duration: "1m30s", max_request_size: "33554432", tls:
"disabled")
               Log Level: info
                   Mlock: supported: false, enabled: false
           Recovery Mode: false
                 Storage: inmem
                 Version: Vault v1.8.0
             Version Sha: 82a99f14eb6133f99a975e653d4dac21c17505c7

...
...
...
WARNING! dev mode is enabled! In this mode, Vault runs entirely in-memory
and starts unsealed with a single unseal key. The root token is already
authenticated to the CLI, so you can immediately begin using Vault.

You may need to set the following environment variable:

    $ export VAULT_ADDR='http://127.0.0.1:8200'

The unseal key and root token are displayed below in case you want to
seal/unseal the Vault or re-authenticate.

Unseal Key: /U//pSIt3TWRDCKK2TGvldlO5TGrSasW1119E2BkYps=
Root Token: s.rLaYfhmgXpa5dWsp0pzTImtR

Development mode should NOT be used in production installations!
```

F. Added Vault address to the environment variable

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

G. Accessed the Vault UI on below mentioned URL with the root token provided by vault server -dev command
http://127.0.0.1:8200/ui/vault/auth?with=token

1. **Setting up AWS secrets engine (Commands executed with received response)**
   Step1:

```
$ vault secrets enable aws
Success! Enabled the aws secrets engine at: aws/
```

Step2:

```
$ vault write aws/config/root access_key=AKIA secret_key=abcdefg
region=us-east-1
Success! Data written to: aws/config/root
```

2. **Vault API command translated from Vault CLI command**

```
curl -X POST -H "X-Vault-Token:s.RK1H0HzqJdrtKwO1X1MI3tul"
"http://127.0.0.1:8200/v1/aws/roles/my-role" -d
'{"credential_type":"iam_user","policy_document":"{"Version":"201
2-10-17","Statement":[{"Effect":"Allow","Action":"ec2:*","Resourc
e":"*"}]}"}'
```

3. **Write a Vault ACL policy that would give you the permissions to run the above command.**

Step1: Login to vault on localhost on: http://127.0.0.1:8200/ui/vault/auth?with=token

Step2: Provide the root token to login

Step3: Navigate to policies tab

Step4: Click on “Create ACL policy” button under ACL Policies

Step5:
a) Enter the name of the policy as “foo” under “Name” field.

b) Paste the following in Policy field:

```
path "aws/roles/my-role" {
    capabilities = ["update"]
}
```

c) Save the policy by Clicking on “Create policy” button.

Step6:

```
$vault token create -policy="foo"
Key                     Value
---                     -----
token                   s.RK1H0HzqJdrtKwO1X1MI3tul
token_accessor          9JwVpu7qgdFn5SQK3oUg5mF5
token_duration          768h
token_renewable         true
token_policies          ["default" "foo"]
identity_policies       []
policies                ["default" "foo"]
```

Learnings:

The ACL policy works with “update” capabilities but does not work with “create” and “read” capabilities for the above command. Not sure why.

Created the policy with only “write” capabilities first - it did not work
Edited the policy again with “read” and “write” capabilities- it did not work
Edited the policy again with “update” capabilities - it did work

Functional: Customer Response to a known issue

1.

Hello Joe,

Good Morning!
Thank you for reaching out to us.

I would like to explain to you how Vault works in your situation.

If you define a policy for "apps/*", the policy would also match "apps/data".
Specifically, when there are potentially multiple matching policy paths, P1 and P2, the following matching criteria is applied:

1. If the first wildcard (+) or glob (*) occurs earlier in P1, P1 is lower priority
2. If P1 ends in * and P2 doesn't, P1 is lower priority
3. If P1 has more + (wildcard) segments, P1 is lower priority
4. If P1 is shorter, it is lower priority

5. If P1 is smaller lexicographically, it is lower priority

For example, given the two paths, "apps/data/*" and "apps/data/+/+/secrets/*", the first wildcard appears in the same place, both end in * and the latter has two wildcard segments while the former has zero. So we end at rule (3), and give "apps/data/+/+/secrets/*" lower priority.

So, you would need to use named paths for the apps and environments.

Also, I am forwarding your request to the product team and they can prioritize* the change accordingly.
*Please keep in mind feature requests depends on the product backlog and might take some time.

Please feel free to reach out to us for any further clarification.

Thanks,
Namisha

Self-managed Vault - Generate-Root API

1.

Hello Mr. XYZ,

Good Afternoon!
Thank you for reaching out to us.

Looking at your query, I would like to tell you that we don't currently have an API for decoding root tokens with OTP. However, the process of decoding the token is a relatively simple XOR of the bytes of the OTP against the bytes of the token, so you can use any language to do so on your side.

I am giving you an example of code written in NodeJS below:

NodeJS:
```
//include module
var xor = require('base64-xor');

//decode takes two parameters: key (utf8), data (base64)
xor.decode('<OTP>','<Encoded token>');
//output: decoded token
```

Please feel free to reach out to us if you face any issues further.

Thanks,
Namisha

Links referred:
https://github.com/hashicorp/vault/issues/5236
https://openbase.com/js/base64-xor/documentation