Secrets Management

Do’s and Don’ts | Peter Gasper | 16.6.2021

LIFE IS FOR SHARING.

Whoami

Peter Gasper

- security engineer
- practicing DevOps at Deutsche Telekom Pan-Net
- currently „Access & Encryption“ squad lead

Slack: cncfsk - #Peter Gasper

e-mail: peter@gasper.cc

Blog: https://malgregator.com

GitHub: https://github.com/viralpoetry

Agenda

- Problems with secrets
- HashiCorp Vault
- Vault‘s journey in Pan-Net
- Vault Open Source limitations
- Conclusion

Problems with secrets

If you are maintaining applications, at some point you have to:

- rebuild infrastructure
- change password
- share credentials
- revoke access

Sensitive data often used during deployment:

- API keys
- SSH credentials
- passwords

Problems with secrets

Secret is anything used for authentication, authorization or encryption:

- *Webserver* (TLS cert, DB credentials, API keys)
- *FreeRADIUS* (shared secret with the VPN HW)
- *Database* (credentials – user/password)

Common problems:

- Sensitive credentials and keys are stored in code repository (GitLab, GitHub, …)
- Sensitive credentials and keys are stored in plain text
- Sensitive credentials and keys are shared in numerous places

Problems with secrets

Traditional approach – small teams:

- PGP
- git-crypt
- ansible-vault

Problems:

- secrets are still committed to a version control repository
- people leaving organization – access to keys/passwords can‘t be revoked – rotate all the secrets
- basically, no lifecycle

Problems with secrets

Solution - infrastructure “password” manager

Basic Requirements:

- single source of truth
- provides API interface
- encryption
- detailed auditing
- ability to revoke access
- multiple authentication methods
- highly available

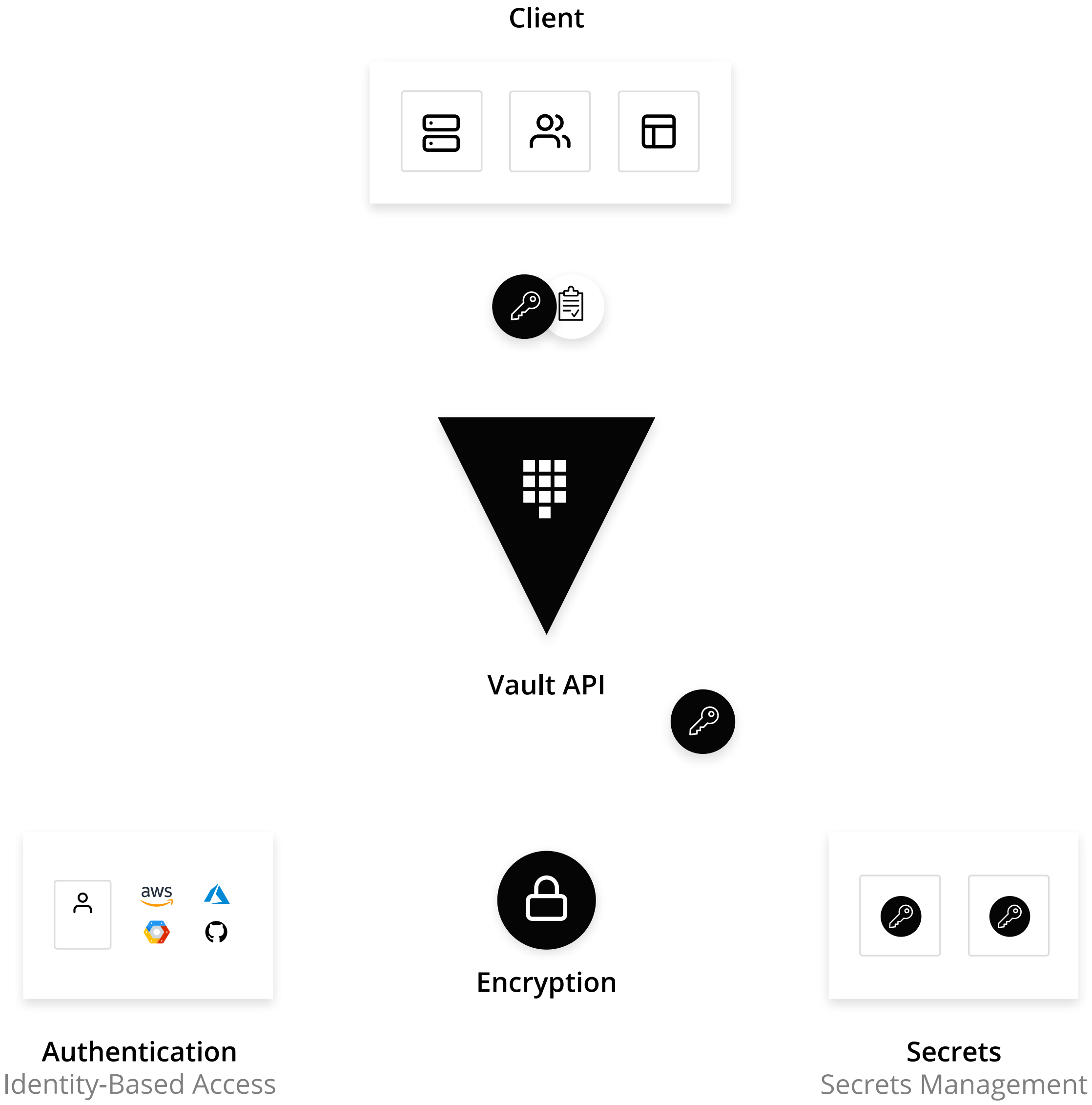
Existing solutions

- Keywhiz by Square
- Confidant by Lyft
- Conjur by CyberArk
- Vault by HashiCorp

Vault by HashiCorp

We chose Hashicorp Vault

- Golang binary
- highly available key/value store
- encryption – Shamir secret sharing scheme
- easy prototyping (vault server -dev)



Vault by HashiCorp

Storage backends – Raft, Consul, Etcd, FoundationDB

Secrets Engines

- **Static** - k/v store for any blob of data – passwords, API tokens etc.
- **Dynamic** - Database credentials, SSH access, AWS, Google Cloud, etc.
- **Encryption** - PKI certificate authority, Transit backend

Auth Methods

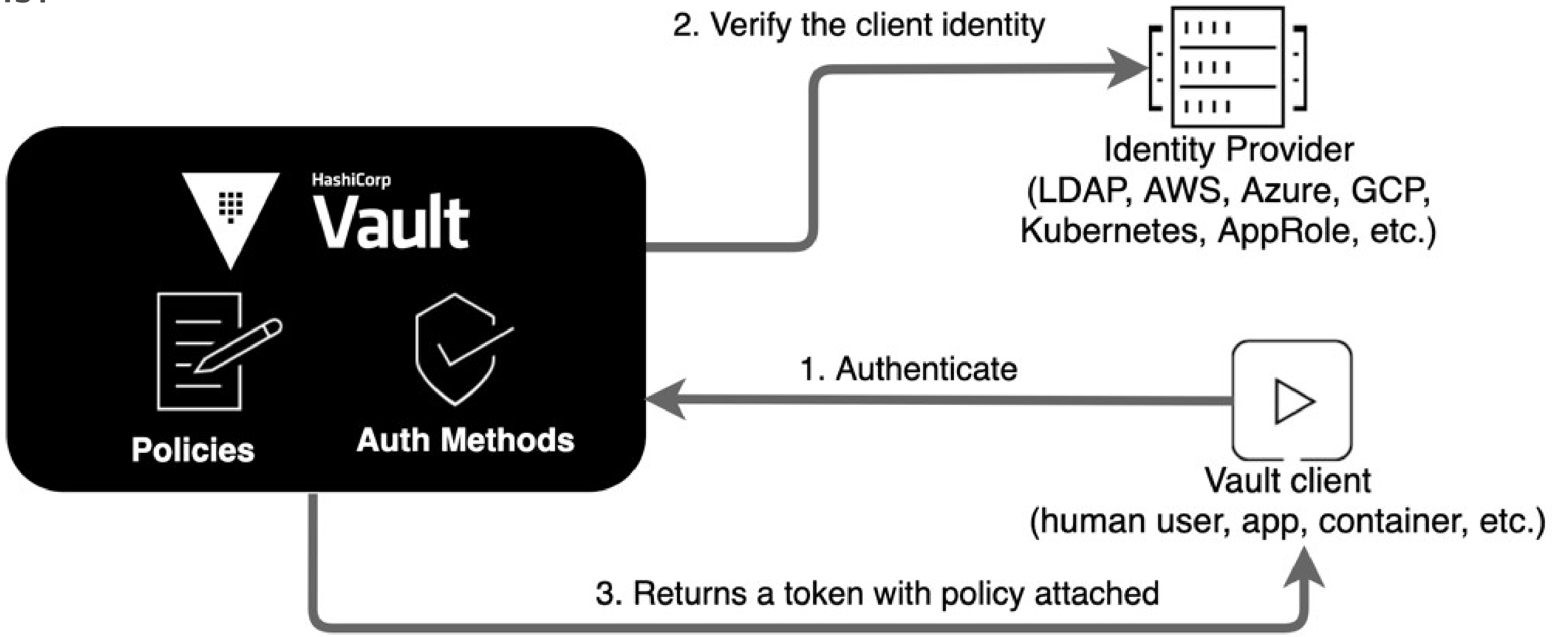
- machine oriented (TLS, JWT, Tokens)
- user oriented (user/pass, LDAP, GitHub, OKTA, Kubernetes, Radius, ...)

Vault by HashiCorp

Every succesful authentication backend results in a **token**.

Every token has **access rights based on defined policy**.

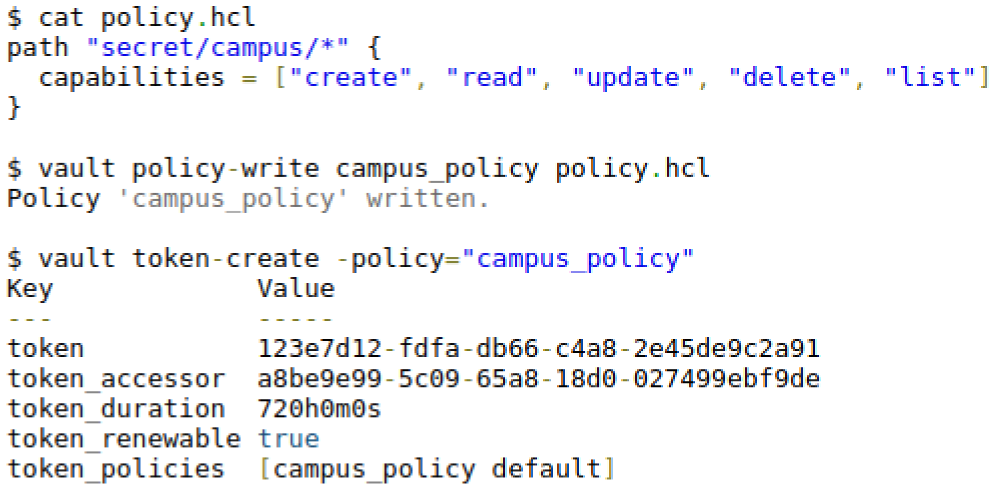
Secrets are **accessed using tokens**.


2. Verify the client identity
Identity Provider
(LDAP, AWS, Azure, GCP,
Kubernetes, AppRole, etc.)
HashiCorp
Vault
Policies
Auth Methods
1. Authenticate
Vault client
(human user, app, container, etc.)
3. Returns a token with policy attached

Vault by HashiCorp

Token:

- has expiration (TTL)
- can be renewed

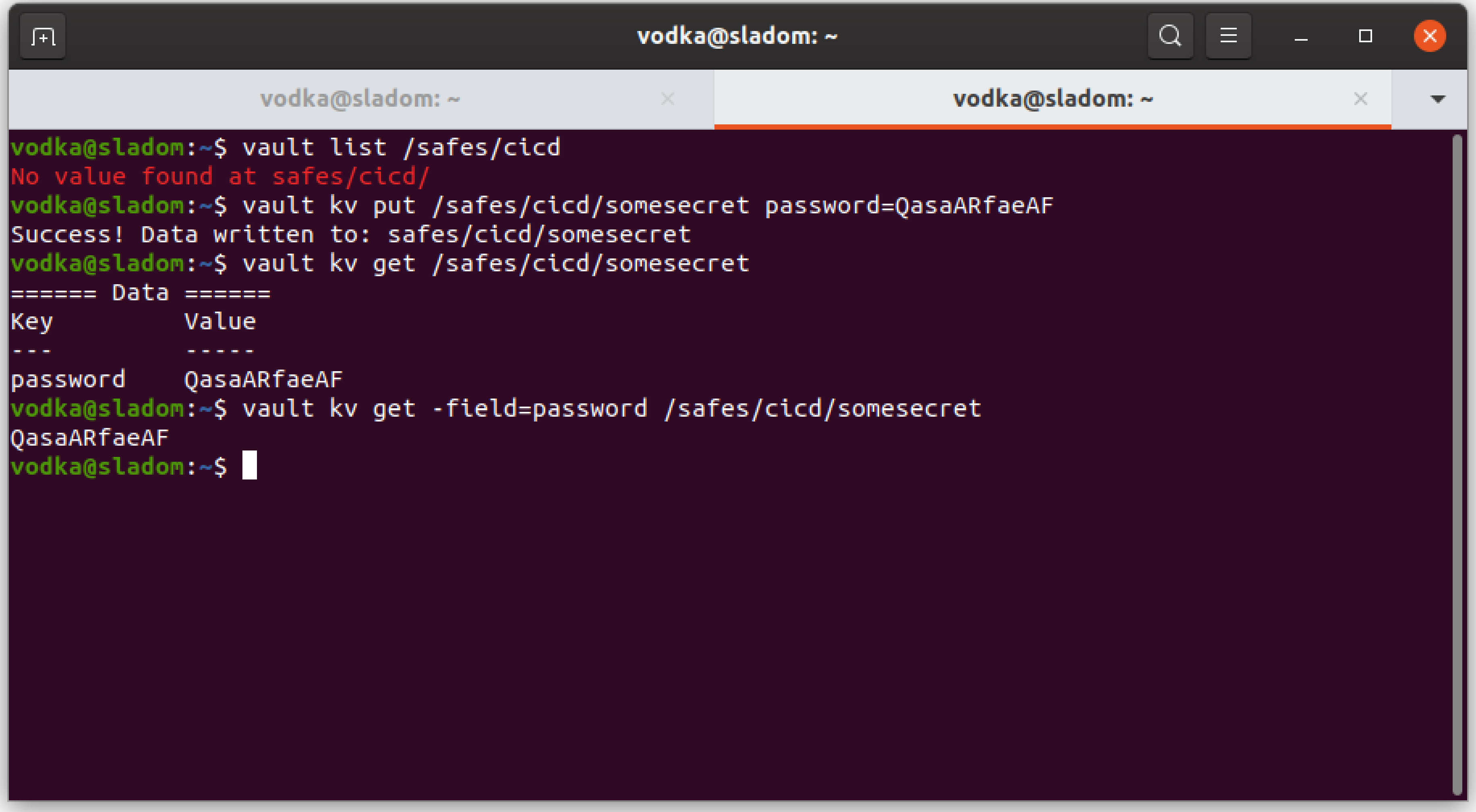


```
$ cat policy.hcl
path "secret/campus/*" {
  capabilities = ["create", "read", "update", "delete", "list"]
}

$ vault policy-write campus_policy policy.hcl
Policy 'campus_policy' written.

$ vault token-create -policy="campus_policy"
Key             Value
---             -----
token           123e7d12-fdfa-db66-c4a8-2e45de9c2a91
token_accessor  a8be9e99-5c09-65a8-18d0-027499ebf9de
token_duration  720h0m0s
token_renewable true
token_policies  [campus_policy default]
```

Vault by HashiCorp - CLI usage



```
vodka@sladom:~$ vault list /safes/cicd
No value found at safes/cicd/
vodka@sladom:~$ vault kv put /safes/cicd/somesecret password=QasaARfaeAF
Success! Data written to: safes/cicd/somesecret
vodka@sladom:~$ vault kv get /safes/cicd/somesecret
====== Data ======
Key         Value
---         -----
password    QasaARfaeAF
vodka@sladom:~$ vault kv get -field=password /safes/cicd/somesecret
QasaARfaeAF
vodka@sladom:~$ 
```

Vault by HashiCorp - Ansible roles

You define **Vault address, mount point, path & name of a secret**, e.g. *radius_vpn_secret.*

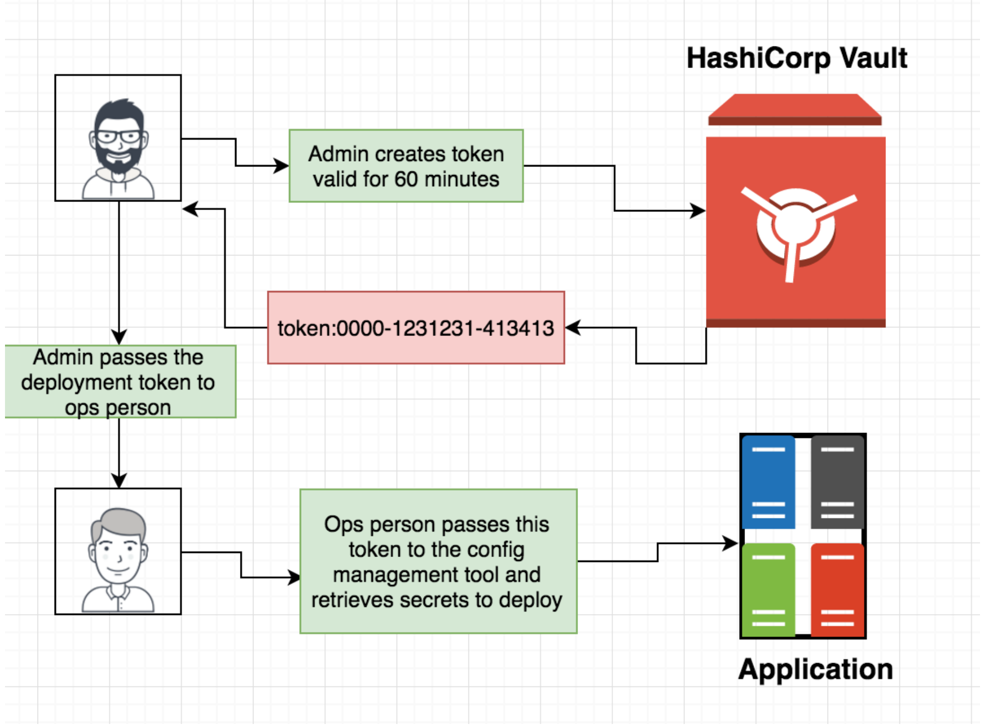
Roles can generate the passwords if it does not exist

```
- hosts: radius-vpn
  gather_facts: no
  become: yes
  vars:
      - vault_mount: ”secret”
      - vault_path: ”my_project”
      - vars_stored:
          - { var: ’ldap_bind_password’ , key: ‘password’ , password: yes }
          - { var: ’radius_vpn_secret’ , key: ‘password’ , password: yes , length: 12 }
  roles:
     - ansible-load-secrets
     - ansible-save-secrets
     - ansible-freeradius
```

Vault‘s journey in Pan-Net

2017 – MVP, basic workflow

- Vault v0.6.5, single instance with filesystem backend, deployed with Ansible
- **Operation**: exchange GPG keys, create policy, issue & deliver tokens
- ansible-load-secrets, ansible-save-secrets roles

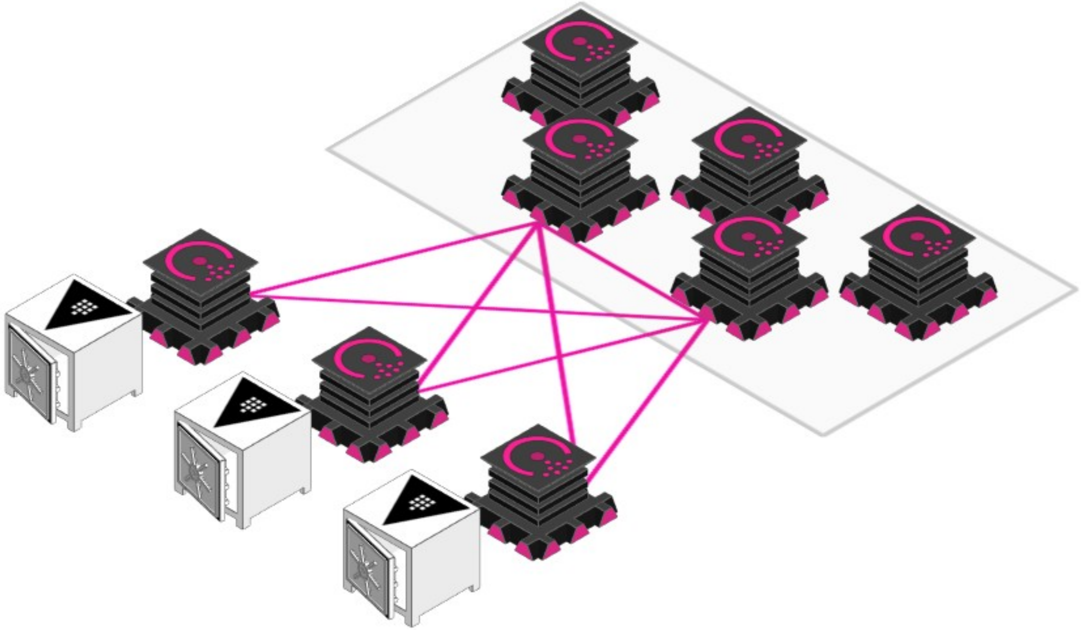


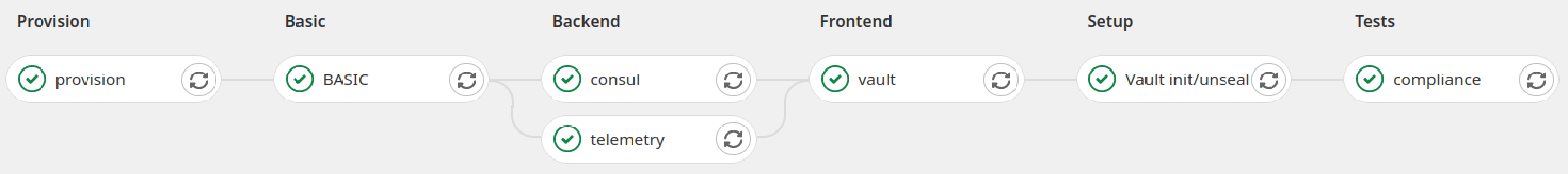
HashiCorp Vault
Admin creates token valid for 60 minutes
token:0000-1231231-413413
Admin passes the deployment token to ops person
Ops person passes this token to the config management tool and retrieves secrets to deploy
Application

Vault‘s journey in Pan-Net

2018 – resiliency, more automation

- whole provisioning & deployment as a code
- access policies provisioned from Gitlab repo
- rolling updates & HA setup with Consul





Provision | Basic | Backend | Frontend | Setup | Tests

provision | BASIC | consul, telemetry | vault | Vault init/unseal | compliance

Vault‘s journey in Pan-Net

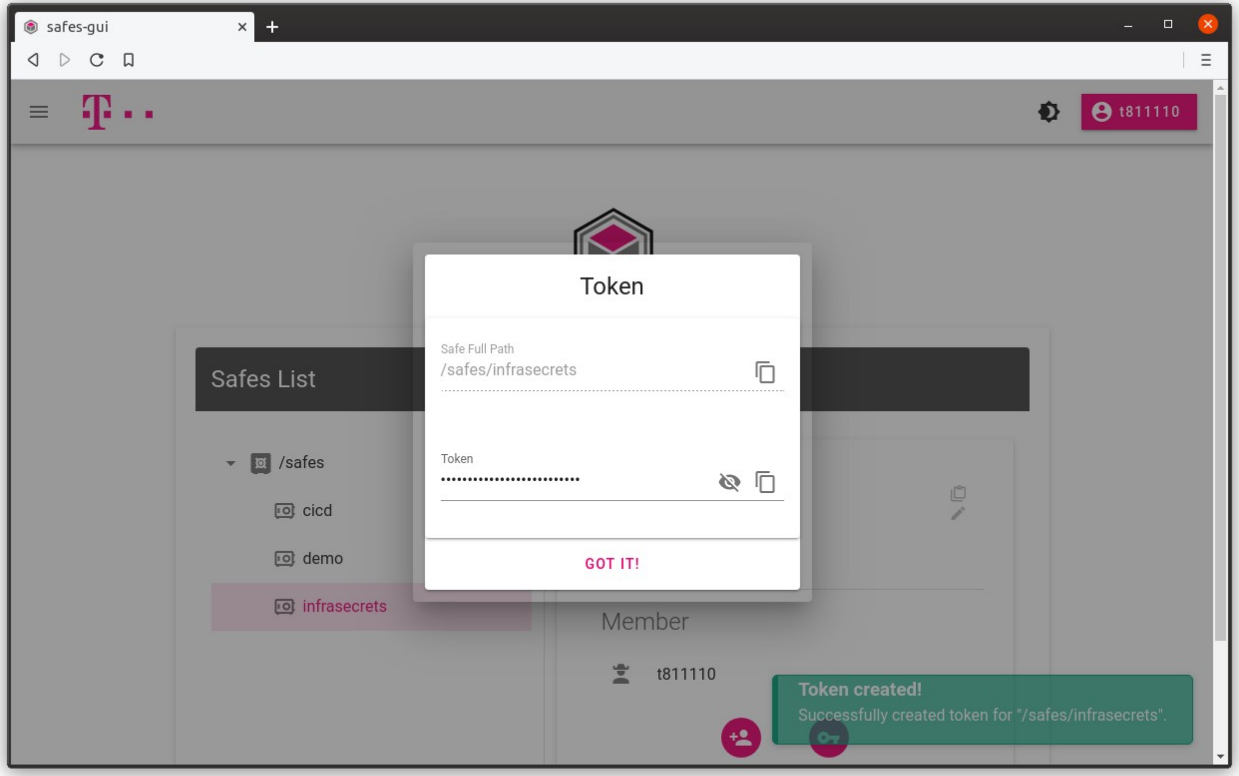
2019 – Self-Service

Due to operation hell, we started designing self-service for the most common use case.

You ask for a “safe“, then you can issue tokens for path and subpaths by yourself.

Modus operandi:

- logged-in person is mapped to an identity
- “Safe“ is Vault identity group
- group has policies
- groups are added/removed to identities



Token

Safe Full Path
/safes/infrasecrets

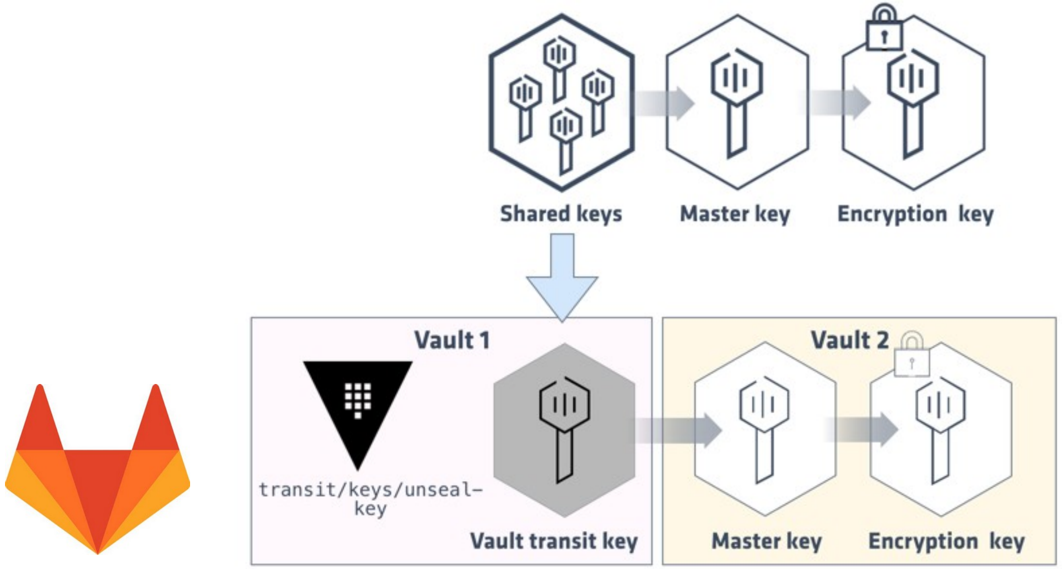
Token
••••••••••••••••••••••••••••••

Token created!
Successfully created token for "/safes/infrasecrets".

Vault‘s journey in Pan-Net

2021 – way more clusters

- migrating to Raft integrated storage
- Gitlab integration using JWT in the pipeline
- multiple new clusters deployed using Helm charts



Shared keys

Master key

Encryption key

Vault 1

transit/keys/unseal-key

Vault transit key

Vault 2

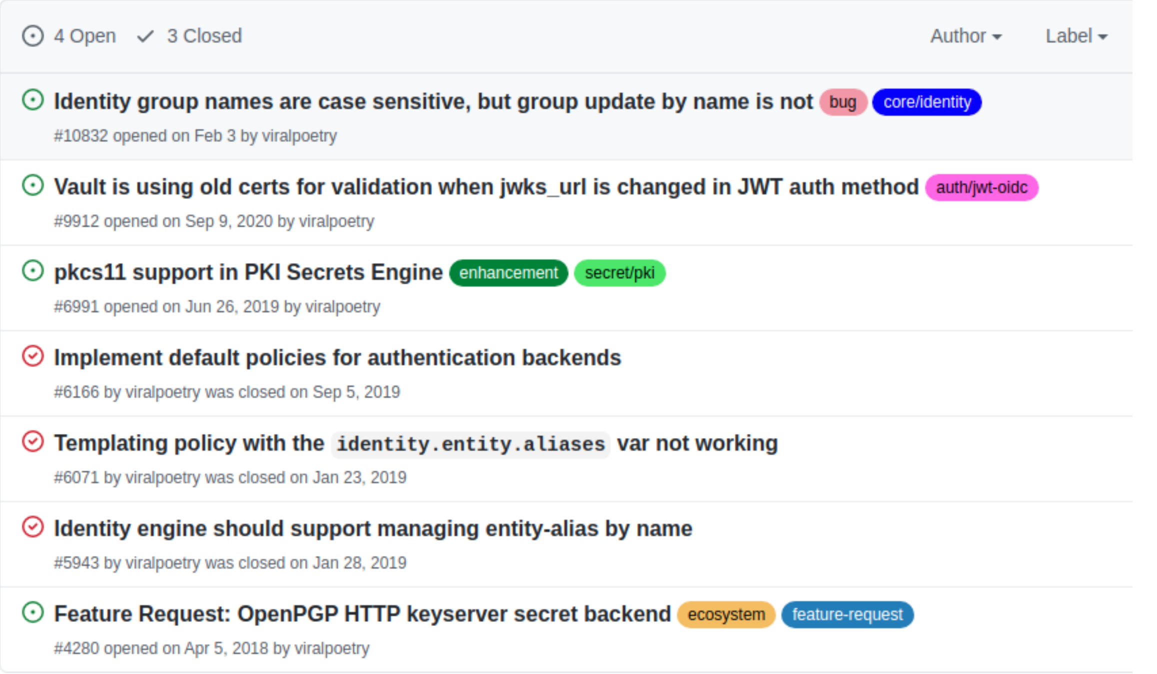
Master key

Encryption key

Vault Open Source limitations

Limitations of the open-source version when doing advanced topics:

- georedundancy
- shamir secret unsealing for the first Vault
- no PKCS #11 support for unsealing nor PKI
- audit backend settings are not propagated in HA



○ 4 Open ✓ 3 Closed Author ▾ Label ▾

○ **Identity group names are case sensitive, but group update by name is not** bug core/identity
#10832 opened on Feb 3 by viralpoetry

○ **Vault is using old certs for validation when jwks_url is changed in JWT auth method** auth/jwt-oidc
#9912 opened on Sep 9, 2020 by viralpoetry

○ **pkcs11 support in PKI Secrets Engine** enhancement secret/pki
#6991 opened on Jun 26, 2019 by viralpoetry

✓ **Implement default policies for authentication backends**
#6166 by viralpoetry was closed on Sep 5, 2019

✓ **Templating policy with the `identity.entity.aliases` var not working**
#6071 by viralpoetry was closed on Jan 23, 2019

✓ **Identity engine should support managing entity-alias by name**
#5943 by viralpoetry was closed on Jan 28, 2019

○ **Feature Request: OpenPGP HTTP keyserver secret backend** ecosystem feature-request
#4280 opened on Apr 5, 2018 by viralpoetry

Conclusion

If I would start again:

- secrets management reflects organizational structure, start with authn/authz lifecycle
- use dynamic secrets engines for new infra
- use policy templates, identity groups, automate role provisioning
- write less Ansible for a setup phase, used libraries like Python HVAC instead

THANKS FOR LISTENING.

Pictures used

https://learn.hashicorp.com/img/vault-auth-basic-2.png

https://learn.hashicorp.com/img/vault-autounseal-12.png

https://easydrawingguides.com/wp-content/uploads/2020/12/Spilt-Milk-Step-10.png