Secret management service for EGI Cloud Infrastructure

Viet Tran (IISAS), Marica Antonacci (INFN), Alvaro Lopez (IFCA)

Motivation

- Applications in EGI infrastructure may need some secrets for deployments and operations (credentials, certificates, passwords, tokens, ...)
  - The secrets are often stored in services codes or configuration files, in clear text
- That implies several security issues:
  - Code repos may not have proper access checks and audits
  - Secrets are difficult to rotate when stored statically in codes
- The secret management service is developed for solving the issue

Requirements

- Secure, industry proven solutions
- High availability, no single point of failure
- Support automation (accessing secrets from VMs in Cloud)
- Usability

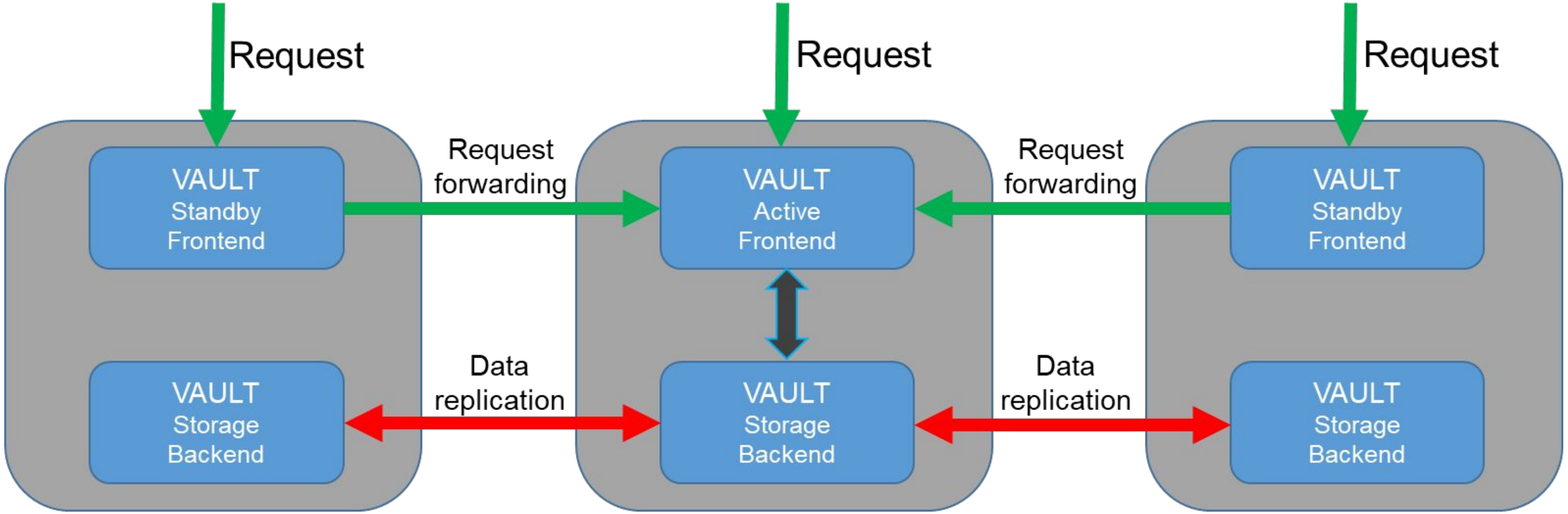
Requirements

- Secure, industry proven solutions
- High availability, no single point of failure
- Support automation (accessing secrets from VMs in Cloud)
- Usability

Solutions

- Hashicorp Vault is well-known
- Three instances, geographically distributed
- Authentication via access tokens what are using to create the VMs
- Universal access point, dedicated client for simple usage

Design of Secret management service



Main endpoint via Dynamic DNS

- Three endpoints, each can serve user requests:
  - https://vault-iisas.services.fedcloud.eu:8200 (IISAS)
  - https://vault-infn.services.fedcloud.eu:8200 (INFN)
  - https://vault-ifca.services.fedcloud.eu:8200 (IFCA)
- Main, universal endpoint https://vault.services.fedcloud.eu:8200 is assigned to IFCA or INFN endpoint via Dynamic DNS
- Cron monitoring and updating main endpoint if needed
- Users can enjoy high availability via the main endpoint without the needs of checking all instances

Dedicated client for usability

- Compatibility is ensured, users can use native Vault client
- However, using Vault native client for accessing Secret management service is uncomfortable:
  - At least two steps needed for every access: login with access token to get Vault's token, then use the Vault's token to access the secrets
  - Setting environments for endpoints, paths and tokens
- A fedcloudclient module is developed for simple usage:
  - Single step for each access
  - Simple syntax: `fedcloud secret get/put/list`
  - Working out of the box without configuration
  - Integration with other services (oidc-agent)

Security reinforcement by client-side encryption

- Users can encrypt secrets using passphrases before uploading to service:
  - 2FA for accessing secrets: token and passphrase
  - Different secrets may have different passphrases: limiting exploits in the case of security breaches
- Client encryption is realized via a simple option `--encrypt-key pass-phrase`

```
$ fedcloud secret put certificate cert=@hostcert.pem --encrypt-key
my-pass-phrase
```

- Decryption is realized via a simple option `--decrypt-key pass-phrase`

```
$ fedcloud secret get certificate cert --decrypt-key my-pass-phrase
```

- Source codes for encryption/decryption is available on GitHub for examination

Summary

Secret management service is not just a deployment of Vault service, it is much more:

- Three servers, geographically distributed for high availability
- Single main endpoint for easy, universal access to the service
- Dedicated client for easy adoption
- Security improvement by client-side encryption

Links

- Secret management service endpoint: https://vault.services.fedcloud.eu:8200/
- Documentation: https://vault.docs.fedcloud.eu/
- Source code of fedcloudclient module: https://github.com/tdviet/fedcloudclient/blob/master/fedcloudclient/secret.py#L159
-

Thank you for your attention

Questions: tdviet@gmail.com