Vancouver HashiCorp Users Group

Alberto Alvarez   Phil Whelan

April 2019

Bench

Agenda

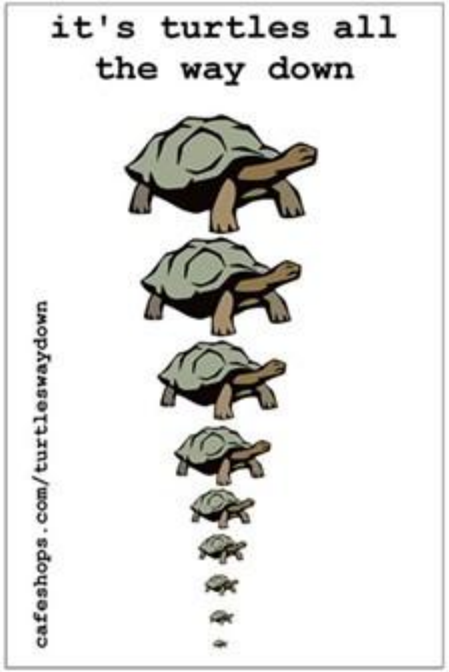
- HashiCorp Vault
- Persistent Credentials
- Ephemeral Credentials
- Vault Database Secrets Engines
- Our Architecture
- Supporting Engineers
- Lessons Learnt
- Future Stuff
- Demo

How To Manage Credentials?

In a config file?

In a database?

How about encrypted?



A Tool For Managing Secrets...

HashiCorp

Vault

Why We Use Vault

Security

Encryption

Policy-based access

Authentication

Integration with SSO, Kubernetes, AWS and more...

Flexibility

Tokens and TTLs galore!

“HashiCorp Vault has changed the way we manage secrets.”

Us, just now
Bench Accounting

Persistent Credentials

It's hard to keep a secret secret forever

Long-lived credentials become less secure with time

Do role specific credentials help?

Ephemeral Credentials

What if credentials expired?

Vault Database Secrets Engine

With a **database secrets engine** plugin, Vault can…

- **Create** credentials based on access **policy**
- Automatically **revoke** credentials based on an **expiry**

Trusting Vault

Requires giving Vault **root** or other high-level database privileges

Root Password Recovery

AWS RDS gives you the ability to reset the root password

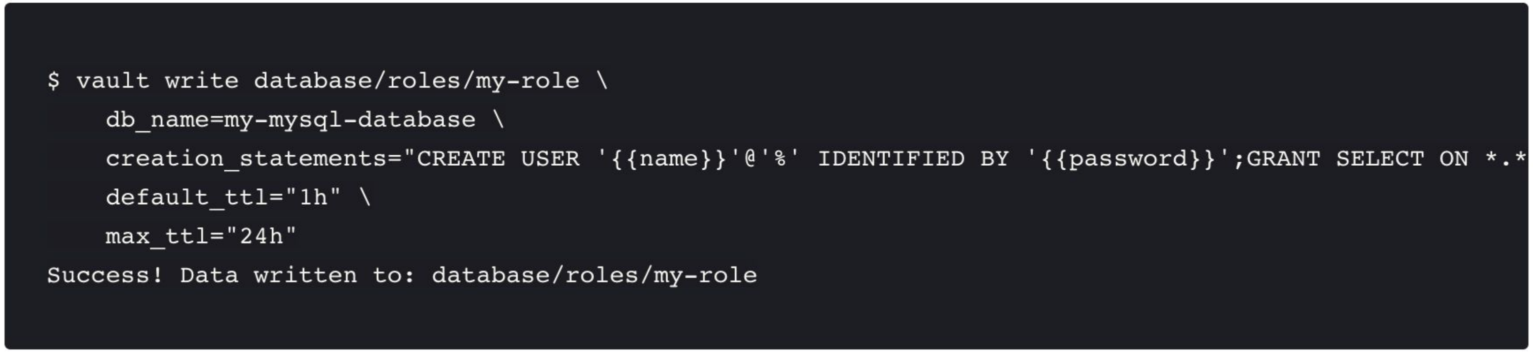
Secrets Engine Setup

Can be done with a few **CLI commands**...

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

```
$ vault write database/config/my-mysql-database \
    plugin_name=mysql-database-plugin \
    connection_url="{{username}}:{{password}}@tcp(127.0.0.1:3306)/" \
    allowed_roles="my-role" \
    username="root" \
    password="mysql"
```

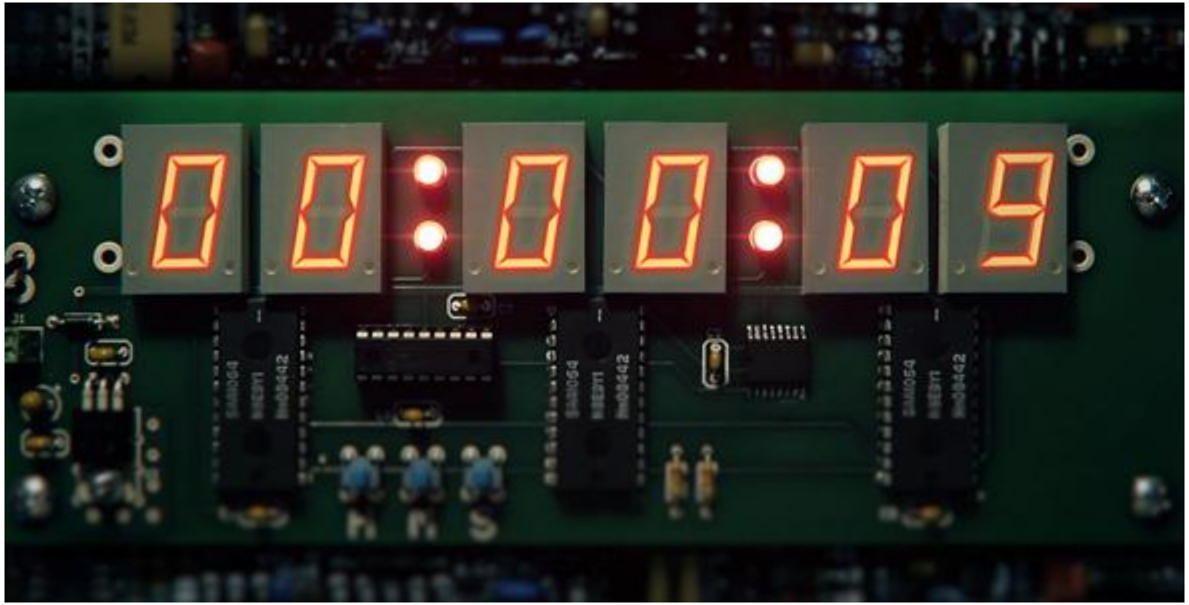
Secrets Engine Setup



```
$ vault write database/roles/my-role \
    db_name=my-mysql-database \
    creation_statements="CREATE USER '{{name}}'@'%' IDENTIFIED BY '{{password}}';GRANT SELECT ON *.*
    default_ttl="1h" \
    max_ttl="24h"
Success! Data written to: database/roles/my-role
```

We use **Terraform** to do this for the **22 databases** we manage this way

Expiring Credentials



“Pop quiz, hotshot. There's a password in your app. Once the app goes to 50 requests per second, the password is critical. In an hour I’m gonna delete the password. What do you do? What do you do?” - Your Vault server

Timely Renewals

Ask for new credentials ***before*** they expire.

You can use

- **Client libraries**

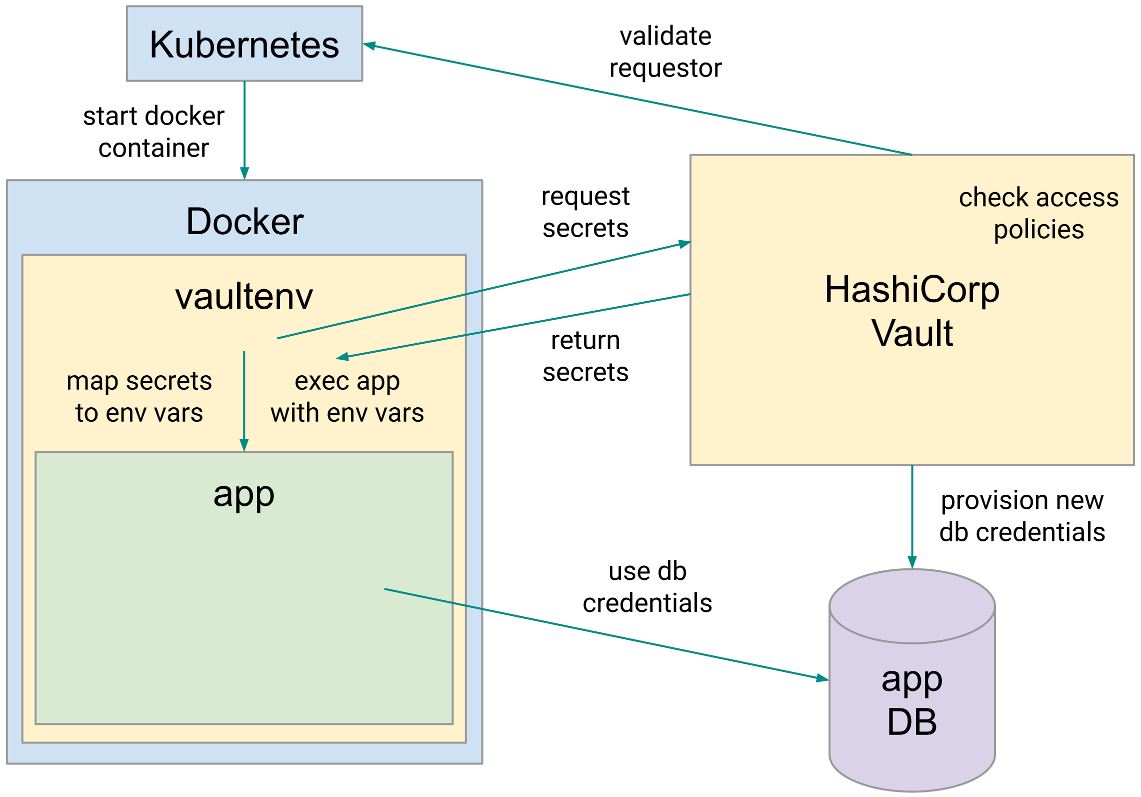
We use

- **Environment variables**

This requires **restarting** the app before the credentials expire.

We run a service that **kills** Kubernetes pods nearing credential expiry.

Application Startup Process



Example vaultenv secrets.conf

```
VERSION 2

MOUNT secret
FOO_KEY=service/myapp/foo#key
FOO_SECRET=service/myapp/foo#secret
BAR_API_USERNAME=service/myapp/bar_api#username
BAR_API_PASSWORD=service/myapp/bar_api#password

MOUNT database
DB_USER=creds/service-myapp#username
DB_PASSWORD=creds/service-myapp#password
```

Engineer Credential Request Process

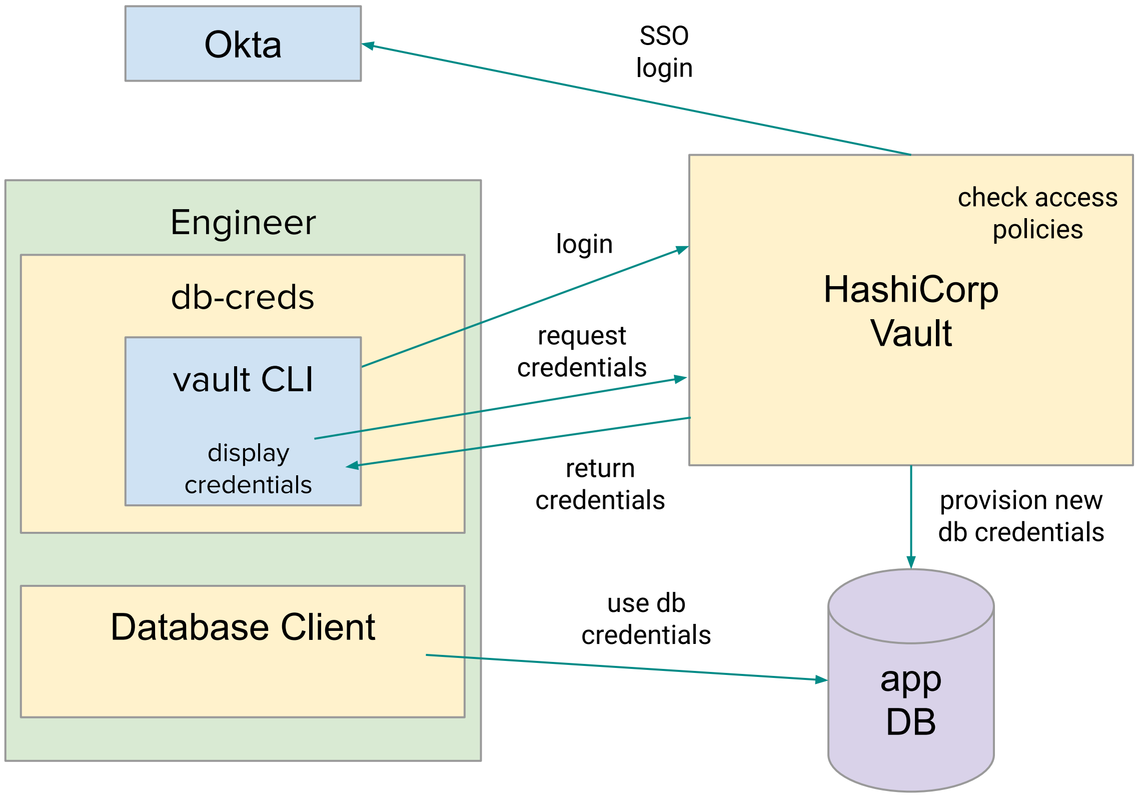
Engineers **self-request access** to any production or non-production database for an appropriate amount of time (defined as a **TTL**).

```
db-creds read my-db

db-creds write my-db

db-creds admin my-db
```

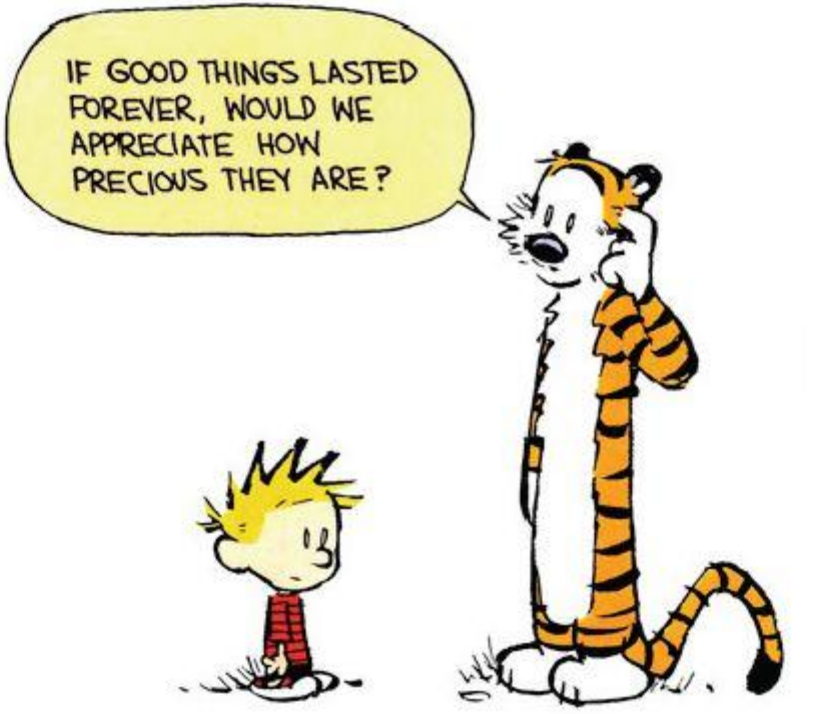
Engineer Credential Request Process



Security vs Convenience

Longer TTLs are less secure

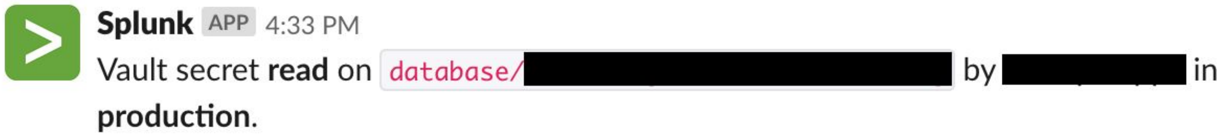
Engineers dislike requesting credentials too often


IF GOOD THINGS LASTED FOREVER, WOULD WE APPRECIATE HOW PRECIOUS THEY ARE?

Auditability

Vault **roles** and **policies** allow us fine-grained access control

Audit log => **Splunk** => **Slack**



Splunk APP 4:33 PM
Vault secret **read** on `database/`[redacted] by [redacted] in **production.**

Visibility into credential usage allows us to

- **Trust** to engineers to self-serve
- **React** to suspicious behaviour

"Доверяй, но проверяй"
Russian proverb
("Trust, but verify")

Lessons Learnt

- MySQL is easier than Postgres
  - Postgres' ownership and role management made it harder to modify the DB schema with ephemeral credentials.
- Provisioning credentials at app startup adds complexity
  - Realized during a minor outage, when we incorrectly thought Vault was at fault
- Opaque TTL hierarchy

Looking To The Future - Part I

“Credentials? Where we’re going, Marty, we don’t need credentials!”
- Crazy old sysadmin / time-traveller

3rd party integration

- Stitch
- AWS Glue
- AWS Data Pipeline

Looking To The Future - Part II

"Credentials? Where we're going, Marty, we don't need credentials!"
- Crazy old sysadmin / time-traveller

Vault rotating root passwords

Review the TTLs we use

- Are Engineers happy? Or too happy?

Other ephemeral credentials?

- e.g. SSH access

Demo

https://git.io/fjslg

Questions