

---

**LET'S LEARN ABOUT  
VAULT**

## VICTORIA (VIKTORIJA) ALMAZOVA

- ▶ Cloud Security Architect
- ▶ motorcycles/running/hiking
- ▶ @texnokot





## WHAT I DELIVER TODAY

- ▶ Share my experience with secrets
- ▶ Give an overview of Vault for successful implementation
- ▶ Practical approach
- ▶ Cool stuff for cool guys and girls



- **VAULT OVERVIEW**
- **IMPLEMENTING VAULT**
- **HANDS-ON SCENARIO**

agenda

## WHY DO WE SPEAK ABOUT IT? REALLY?

- ▶ everyone agrees that storing keys in repository is **BAD**
- ▶ everyone knows that something can be done
- ▶ 100 500 solutions available
- ▶ a chicken and egg situation
- ▶ no one is doing anything

## STILL... BUT WHY?

- ▶ priority... or no priority
- ▶ takes time to implement and maintain
- ▶ habits thing?

**OK...**

**BUT WHAT'S THE PROBLEM?**

**audience**



filename:.dockercfg auth email

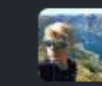


[Pull requests](#)

[Issues](#)

[Marketplace](#)

[Explore](#)



[Repositories](#)

721

**Code**

252

[Commits](#)

68K

[Issues](#)

596K

[Marketplace](#)

[Topics](#)

3

[Wikis](#)

11K

[Users](#)

### Languages

|            |    |
|------------|----|
| Go         | 30 |
| Shell      | 7  |
| JSON       | 5  |
| JavaScript | 3  |
| HTML+ERB   | 2  |
| Python     | 2  |
| Text       | 2  |
| PHP        | 1  |

[Advanced search](#)

[Cheat sheet](#)

## 252 code results

Sort: Best match ▾



[WaleedAbdelmobdi/codeship](#) – dockercfg

Showing the top two matches Last indexed on 7 Jul

```
1  {
2      "auths": {
3          "https://index.docker.io/v1/": {
4              "auth": "d2FsZWVhYkZkZWxtb2JkaTpBc2RAMTIzNA==",
5              "email": "wm@marketscape.com"
6          }
7      }
```



[smenon78/tricorder-appliance](#) – dockercfg

Showing the top two matches Last indexed on 11 Jul

```
1  { "http://10.201.125.67:5000": { "auth": "", "email": "jusilvei@cisco.com" } }
```



[bpolge/dotfiles](#) – dockercfg

Showing the top two matches Last indexed on 2 Jul

```
1  {
2      "https://docker-artifacts.ua-ecm.com": {
3          "auth": "bnBtLWd1eTpBUERCZ1RDYlExbjN0UUZ4",
4          "email": "tcrider@underarmour.com"
5      }
6  }
```





filename:.bash\_profile aws

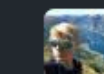


Pull requests

Issues

Marketplace

Explore



Repositories

85K

Code

862+

Commits

1M

Issues

340K

Marketplace

Topics

492

Wikis

Users

Languages

Shell

HTML+ERB

Text

HTML

HTML+Djang

Markdown

1

Showing 863 available code results ?

Sort: Best match ▾



amandazhang/mybash – .bash\_profile

Shell

Showing the top four matches Last indexed on 1 Jul

```
5 PATH="$HOME/Library/Python/2.7/bin:${PATH}"
6 export PATH
```



dancesnitch/workhorse-app – .bash\_profile

Shell

Showing the top four matches Last indexed on 10 Jul

```
1 require 'aws-sdk'
2
3
4 export AWS_ACCESS_KEY_ID=AKIAIB5TQBGYF4ZE2KEQ
5 export AWS_SECRET_ACCESS_KEY=Z5koTXd0MeKIqqt re0u0hLzDWTznHq8NpPrRwjxa
6 export AWS_REGION=us-east-2
7 export S3_BUCKET=workhorseapp
```

Showing the top five matches Last indexed 5 days ago

```
99 complete -F _ssh ssh
100
101 ## AWS
102 AWS_COMP="$(which aws_completer)"
103 if [ -n "$AWS_COMP" ]; then
104     complete -C "$AWS_COMP" aws
105 fi
```

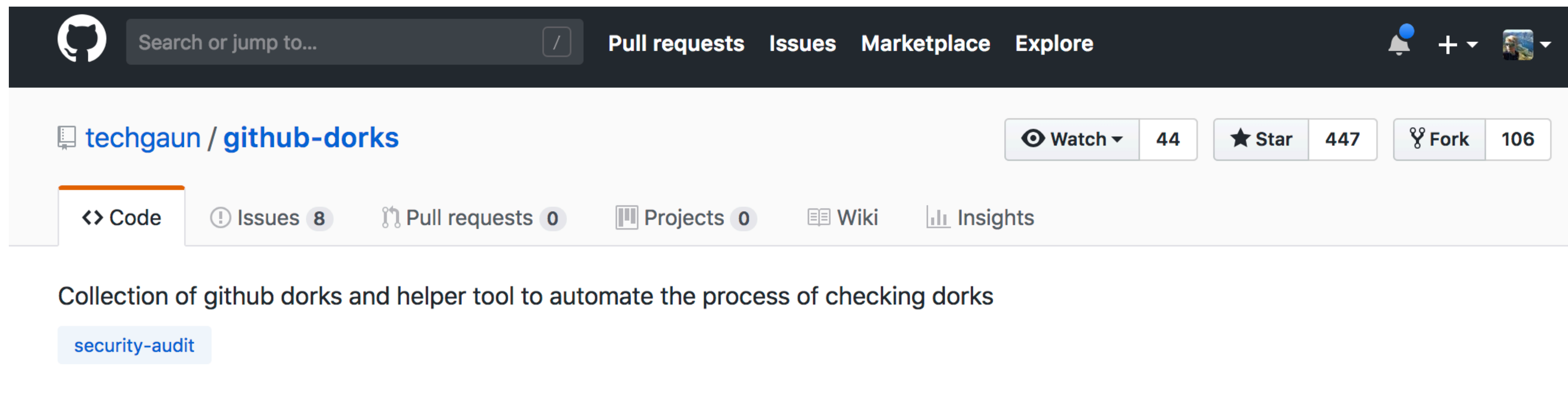
Advanced search

Cheat sheet

LET'S LEARN ABOUT VAULT TEXT

---

# AND IT IS EVEN EASIER NOW...



The screenshot shows the GitHub interface for the repository `techgaun / github-dorks`. The top navigation bar includes the GitHub logo, a search bar, and links for Pull requests, Issues, Marketplace, and Explore. On the right, there are notification and user profile icons. Below the repository name, there are buttons for Watch (44), Star (447), and Fork (106). A secondary navigation bar shows tabs for Code (selected), Issues (8), Pull requests (0), Projects (0), Wiki, and Insights. The repository description reads: "Collection of github dorks and helper tool to automate the process of checking dorks". A tag labeled "security-audit" is visible below the description.

techgaun / github-dorks

Watch 44 Star 447 Fork 106

<> Code Issues 8 Pull requests 0 Projects 0 Wiki Insights

Collection of github dorks and helper tool to automate the process of checking dorks

security-audit

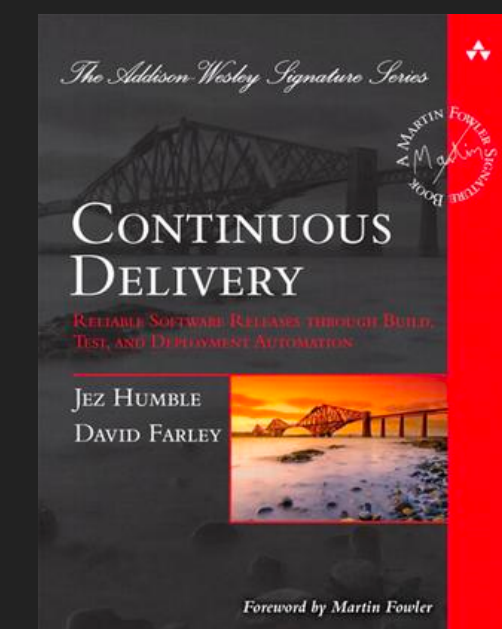
<https://github.com/techgaun/github-dorks>

**“DON'T CHECK PASSWORDS INTO SOURCE CONTROL OR HARD-CODE THEM IN YOUR APPLICATION.**

**OPERATIONS STAFF WILL REMOVE YOUR EYES WITH A SPOON IF THEY CATCH YOU DOING THIS.**

**DON'T GIVE THEM THE PLEASURE.”**

Chapter 2 of the Continuous Delivery: Reliable Software Releases Through Build, Test, And Deployment Automation



## CASES

- ▶ new employee - an access to the repository, to the code, to the creds
- ▶ employee leaves - knew all creds, they aren't changed, sometimes an access isn't removed because of integrations
- ▶ employee asks an access to production DB just to fix the "thing"



**OK...**

**WHAT WE WANT?**

**audience**

## REQUIREMENTS

- ▶ Centrally managed secure secret storage
- ▶ Flexible ACLs for teams, environments, processes
- ▶ Integration and support of tools used in our development
- ▶ Nice and handy UI/CLI
- ▶ And all this we want SECURE

**THERE ARE PLENTY  
SOLUTIONS... AREN'T?**

**audience**

## GIT-CRYPT

- ▶ No central place
- ▶ No ACL
- ▶ Yes. We can integrate with tools
- ▶ Yes. secure

GIT-CRYPT ENABLES TRANSPARENT ENCRYPTION AND DECRYPTION OF FILES IN A GIT REPOSITORY. FILES WHICH YOU CHOOSE TO PROTECT ARE ENCRYPTED WHEN COMMITTED, AND DECRYPTED WHEN CHECKED OUT. GIT-CRYPT LETS YOU FREELY SHARE A REPOSITORY CONTAINING A MIX OF PUB

github



## GIT-SECRETS

- ▶ Almost the same as git-crypt
- ▶ But has ACL

GIT-SECRET IS A BASH TOOL TO STORE YOUR PRIVATE DATA INSIDE A GIT REPO. HOW'S THAT? BASICALLY, IT JUST ENCRYPTS, USING GPG, THE TRACKED FILES WITH THE PUBLIC KEYS OF ALL THE USERS THAT YOU TRUST. SO EVERYONE OF THEM CAN DECRYPT THESE FILES USING ONLY

github

## PASS

- ▶ Unix way
- ▶ Can be integrated with tools
- ▶ Plugins for ACL
- ▶ yeah, but...

Password management should be simple and follow Unix philosophy. With pass, each password lives inside of a gpg encrypted file whose filename is the title of the website or resource that requires the password.

github

## AND OTHERS...

- ▶ Chef - databags
- ▶ Docker - secrets
- ▶ Puppet - hierra
- ▶ Cloud providers with their own solutions
- ▶ Solution X - also has smth
  
- ▶ But....

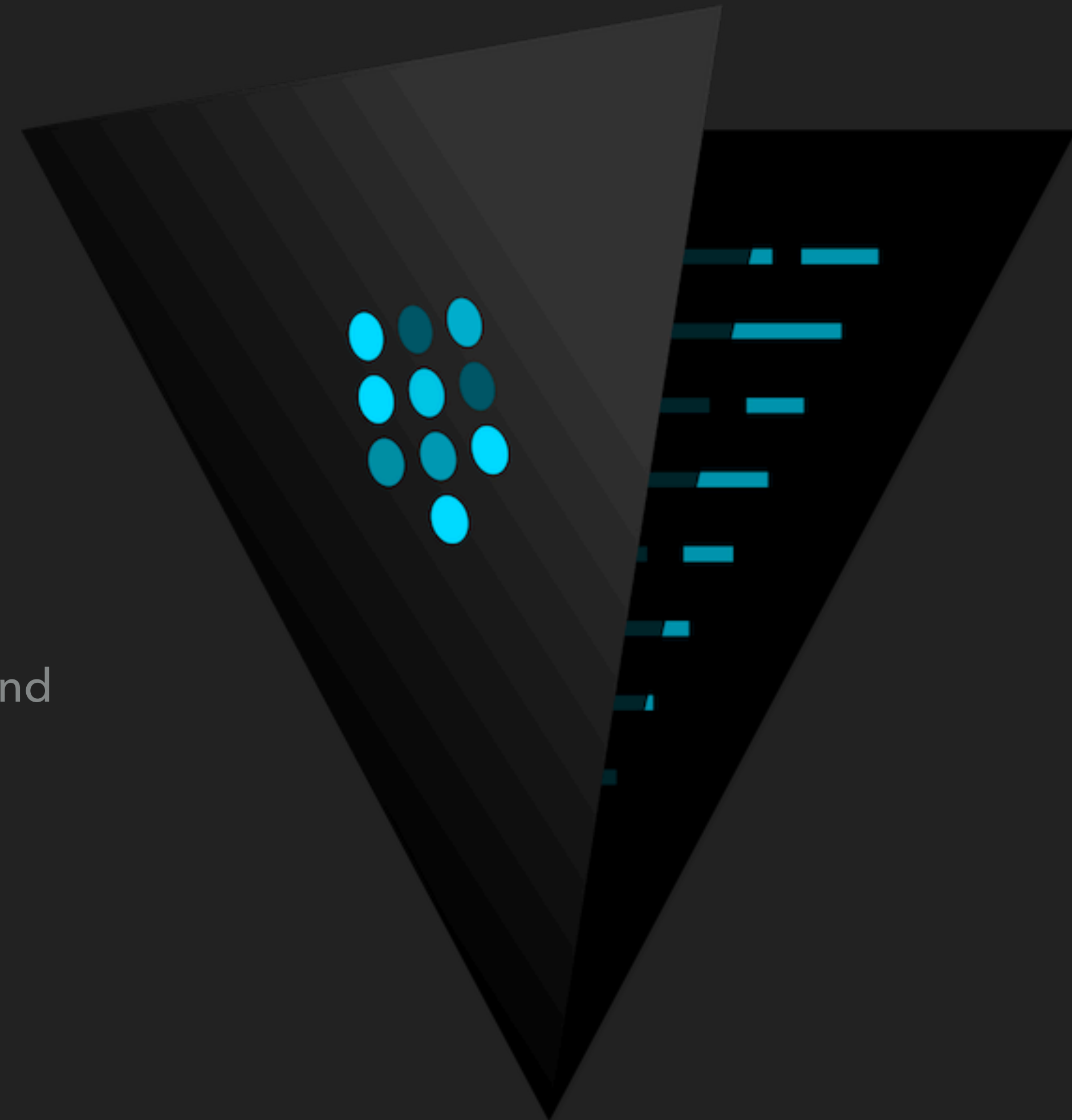
**FINALLY, SHE SPEAKS  
ABOUT VAULT...**

**audience**

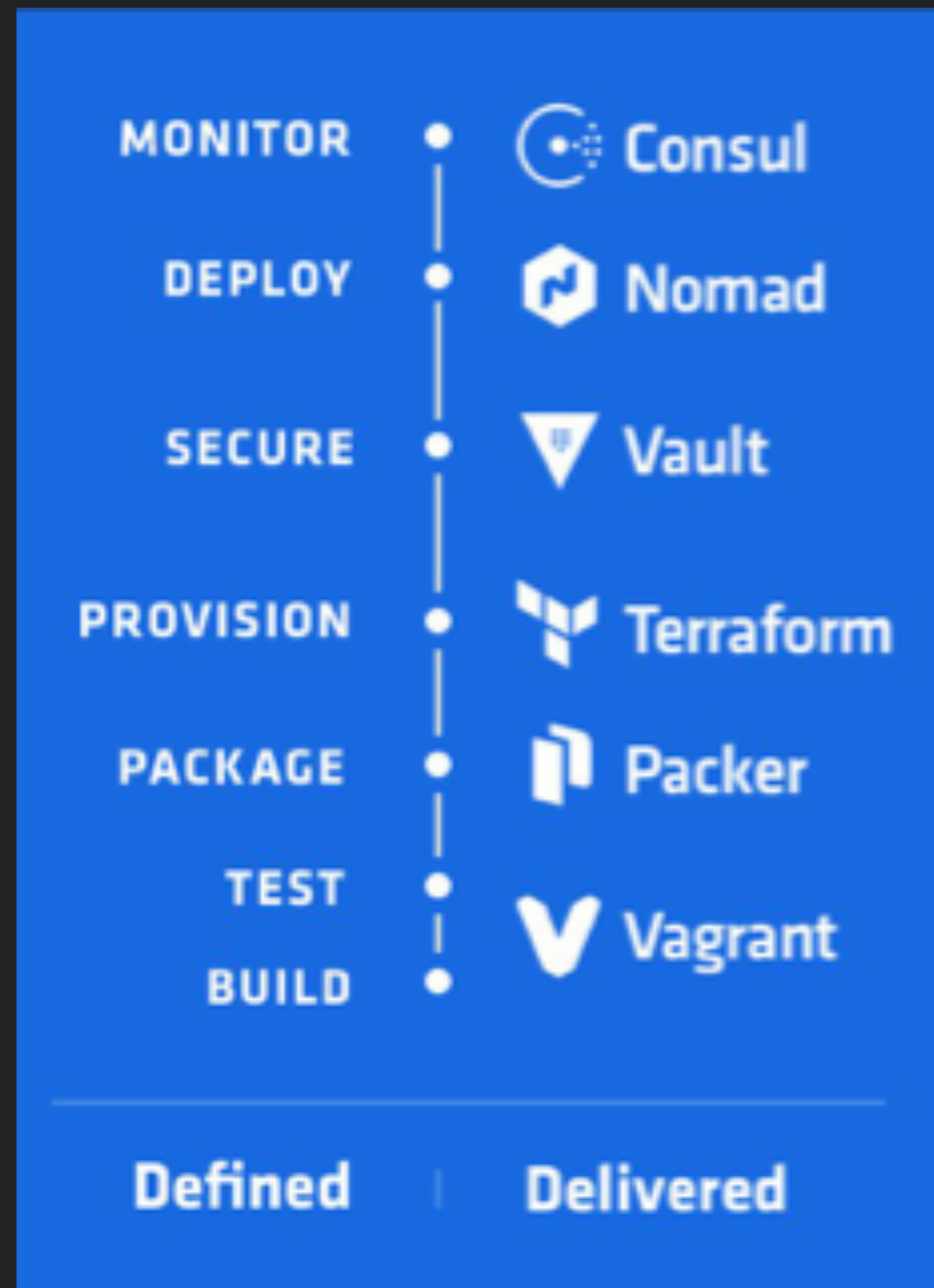


# HASHICORP VAULT

- ▶ Open source
- ▶ Written in Go
- ▶ Configured using HCL
- ▶ Self hosted secure secret storage
- ▶ Self driven architecture
- ▶ Dozens plugins and modules for all tools and languages
- ▶ JSON API
- ▶ 9980 stars on Github
- ▶ More than 1500 forks



## VAULT IN HASHICORP FAMILY



## WHAT WE NEEDED?

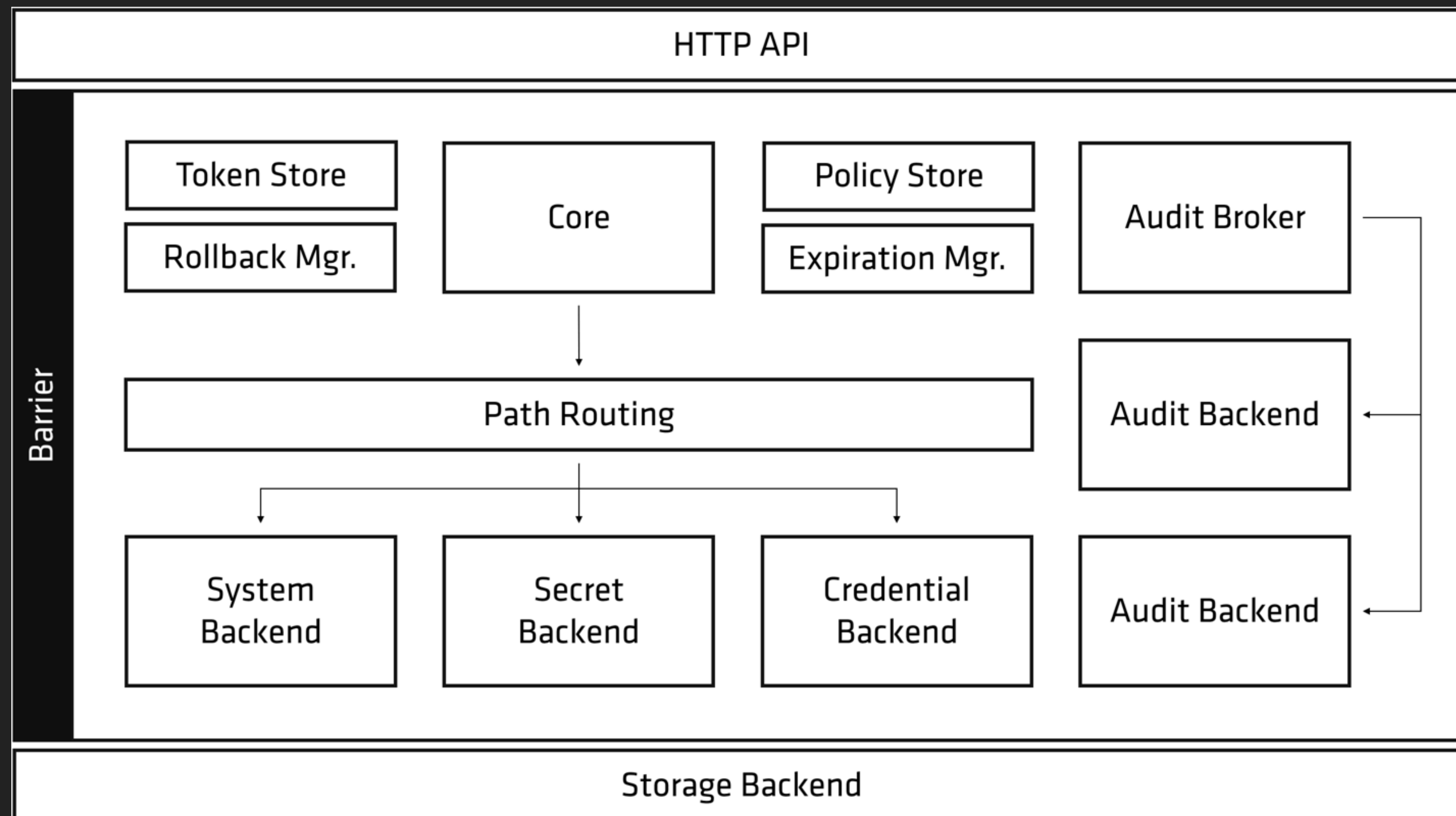
- ▶ Centrally managed secure secret storage
- ▶ Flexible ACLs for teams, environments, processes
- ▶ Integration and support of tools used in our development
- ▶ Nice and handy UI/CLI
- ▶ And all this we want SECURE

## ADDITIONALLY WE GET

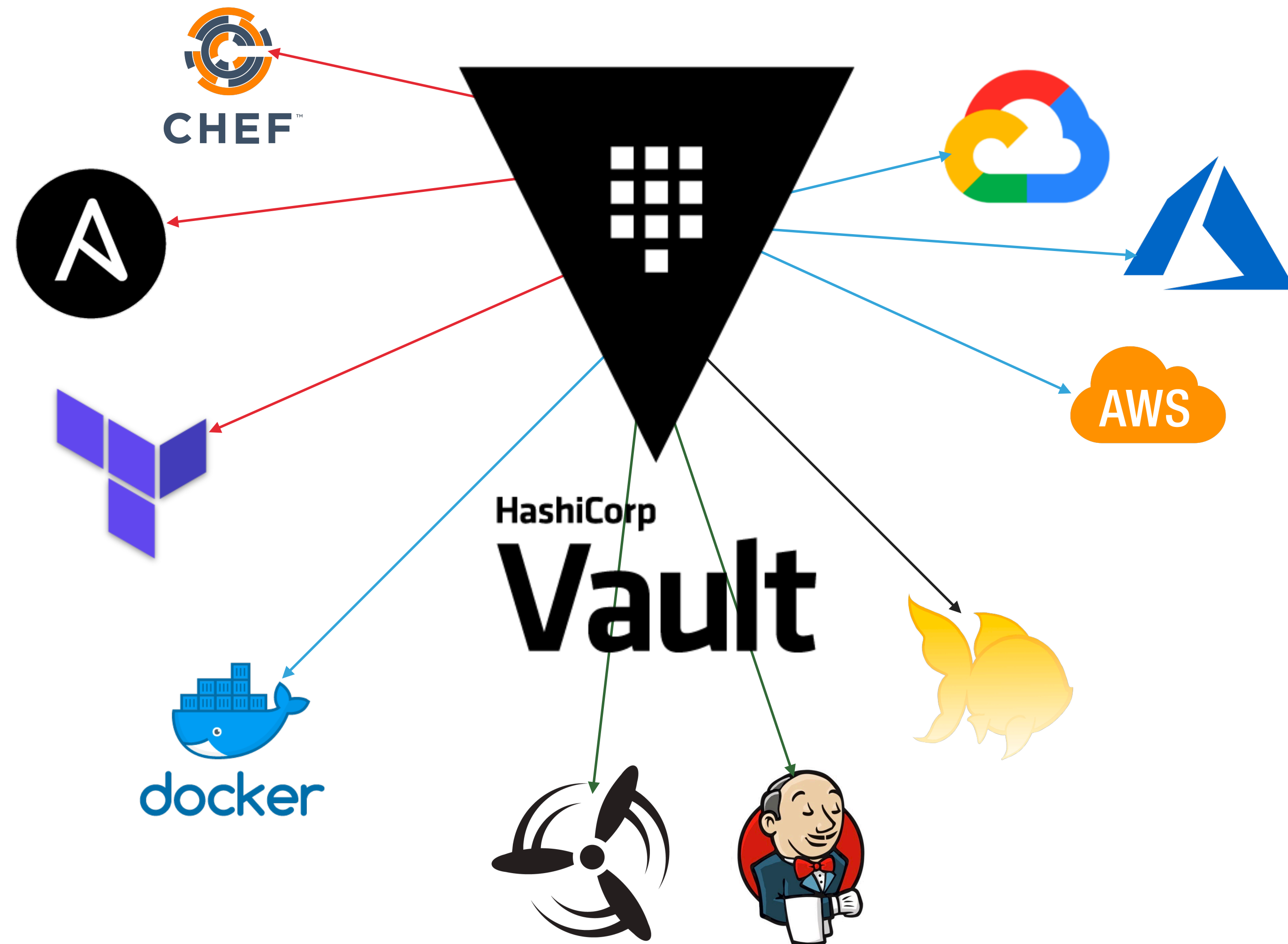
- ▶ Dynamic Secrets
- ▶ Data Encryption
- ▶ Leasing and Renewal
- ▶ Revocation



# HIGH-LEVEL ARCHITECTURE AND MAIN COMPONENTS



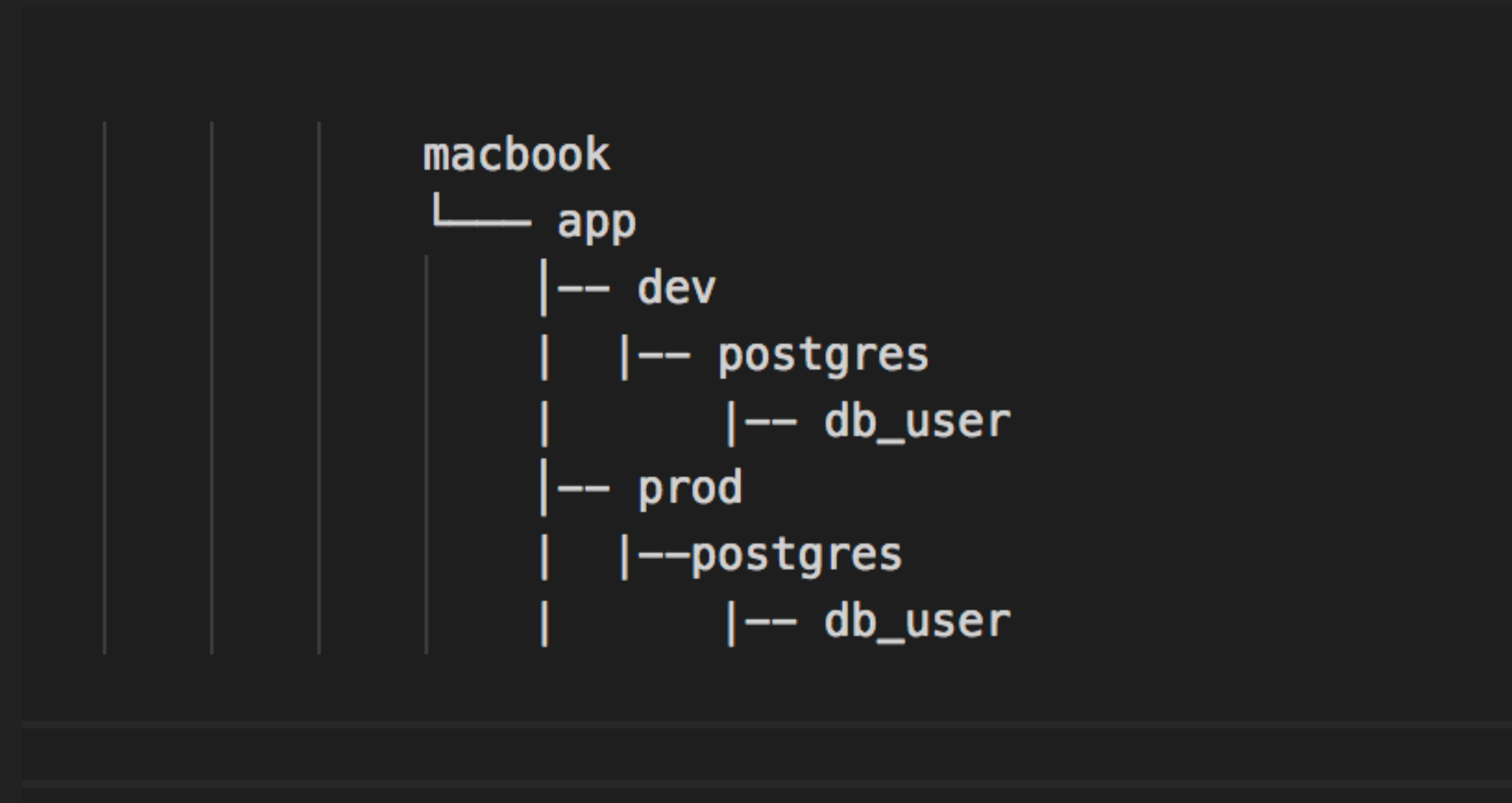
## EXAMPLE OF INTEGRATIONS/ARCHITECTURE



## VAULT BACK END: SECRETS ENGINE

- ▶ Vault behaves similarly to a virtual filesystem
- ▶ Tree structure
- ▶ Secrets Engines:  
AD, AWS, Azure, Consul, Cubbyhole, Databases, Google Cloud, Key/Value, Identity, Nomad, PKI, RabbitMQ, SSH, TOTP

## VAULT BACK END: VISUALISATION (KV CASE)



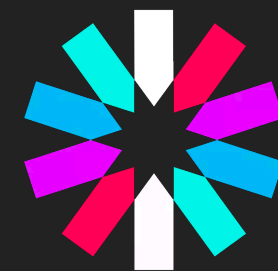
- ▶ \$company/\$application/\$environment/\$service/secret

## VAULT BACK END: DYNAMIC SECRETS

- ▶ Generated when they are accessed
- ▶ Can be revoked immediately after use

## VAULT BACK END: AUTHENTICATION

- ▶ Has pluggable auth methods
- ▶ Different auth mechanisms ends into single "Vault token"
- ▶ Best practices:  
Github, LDAP, AppRole, AWS, Azure, Google Cloud,  
JWT/OIDC, Kubernetes, Okta, RADIUS, TLS certs





## VAULT BACK END: POLICIES

- ▶ Authorisation part: policies sets what a user can access
- ▶ Same format (HCL, but it is JSON compatible)
- ▶ Built-in policies: root and default
- ▶ Policies default: deny
- ▶ Format command

```
# Normal servers have version 1 of KV mounted by default, so will need these
# paths:
path "secret/*" {
  capabilities = ["create"]
}
path "secret/foo" {
  capabilities = ["read"]
}

# Dev servers have version 2 of KV mounted by default, so will need these
# paths:
path "secret/data/*" {
  capabilities = ["create"]
}
path "secret/data/foo" {
  capabilities = ["read"]
}
```

## USE CASE: ANSIBLE

- ▶ Enable plugin
  - ▶ <https://github.com/jhaals/ansible-vault>
- ▶ Syntax (\*.yml, \*.j2):
  - ▶ `{{ lookup('vault', 'macpaw/setapp/dev/mongo/db_user').username }}`

## USE CASE: CHEF

- ▶ Enable plugin
  - ▶ <https://github.com/hashicorp/vault-ruby>
- ▶ Syntax

```
creds = Vault.logical.read("macpaw/setapp/dev/mongo/db_user")

template "/etc/mongodb.conf" do
  source "mongodb.erb"
  variables(
    :username => creds.data[:username],
    :password => creds.data[:password],
  )
end
```

## USE CASE: TERRAFORM

- ▶ ~~Enable plugin~~
- ▶ Syntax

```
creds = Vault.logical.read("macpaw/setapp/dev/mongo/db_user")

template "/etc/mongodb.conf" do
  source "mongodb.erb"
  variables(
    :username => creds.data[:username],
    :password => creds.data[:password],
  )
end
```



## USE CASE: IN APP

```
import os
import hvac

def sample():
    client = hvac.Client(url=os.environ['VAULT_ADDR'],
                        token=os.environ['VAULT_TOKEN'])

    secret = client.read('macpaw/setapp/dev/mongo/db_user')
    return(secret)
```

**HMM, WHAT ABOUT...**

**A CHICKEN AND EGG PROBLEM?**

**audience**



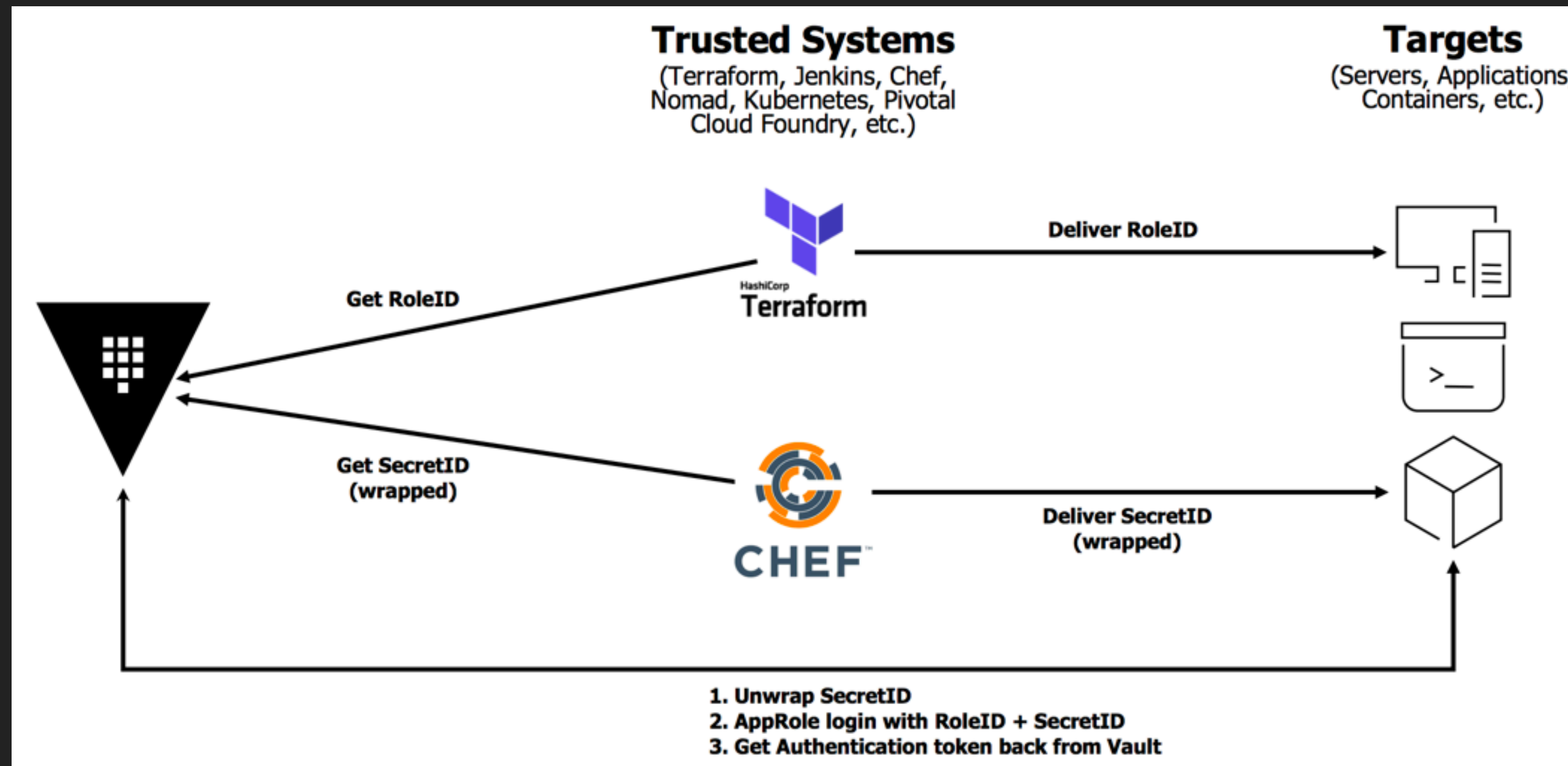
## SECURE INTRODUCTION

- ▶ Cloud native auth capabilities are enough:
  - ▶ AWS, Google Cloud, Kubernetes
  - ▶ <https://www.hashicorp.com/blog/brokering-cloud-identity>
- ▶ What if trust is not enough:
  - ▶ Multiple services on same instance (aka EC2)
  - ▶ Private clouds
  - ▶ More granular approach
  - ▶ Do we trust CI/CD?

## APPROLE?

- ▶ Username and password for machines (Teddy Sacilowski)
- ▶ RoleID - an identifier
  - ▶ Not a secret/sensitive
- ▶ SecretID - unique and secret
- ▶ RoleID + SecretID = Vault Token

## APPROLE IN USAGE



<https://www.vaultproject.io/guides/identity/approle-trusted-entities.html>

<https://github.com/hashicorp/vault-guides>

## VAULT FROM SECURITY EYES

- ▶ There is no mutual trust between the Vault client and server
- ▶ No auth - no access
- ▶ Data is encrypted (Two-man concept for unsealing using Shamir's Secret Sharing technique)
- ▶ Dynamic secrets, totp and etc
- ▶ Break glass procedure

**OK... SOLD IT**

**WHAT NEXT FOR ME?**

**audience**

## VAULT ENTERPRISE

- ▶ Open-source is not enough if needed:
  - ▶ ACL becomes too complicated then Sentinel is needed
  - ▶ Even more performance: Performance Standby Nodes
  - ▶ Even more security: HSM
  - ▶ Tenants usage: Namespaces



## AFTER MEETUP I CAN:

- ▶ Implement Hashicorp Vault in my solutions
- ▶ Share and use it with my team and company
- ▶ Use all benefits and "cookies"
- ▶ Write down Hashicorp Vault in my LinkedIn
- ▶ Make my IaC best!



---

**MAY THE VAULT POWER  
BE WITH YOU**