

PRAQMA

Workshop

Manage your Cloud Native Secrets with Vault
With Henrik Høegh



What is HashiCorp Vault?

- **We need to manage secrets**

- **The lifecycle of secrets**

- When to create, upgrade and delete secrets in our system

- **Storage of secrets**

- Where do we store our secrets in a secure way?
 - How do we pass them on to our environment without exposing it?

- **Access to secrets**

- Who and what have access to our secrets
 - How do we manage access

Why?

ServiceAccount

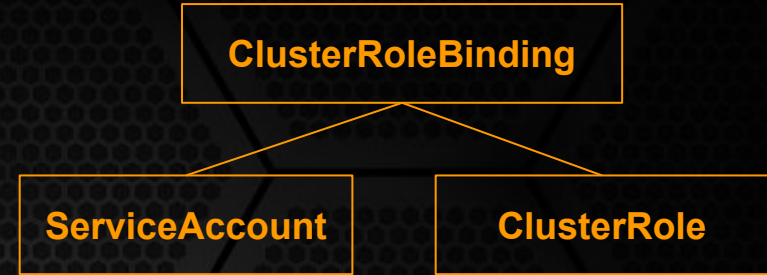
An identity for processes.

ClusterRole

A cluster-wide set of permissions.

ClusterRoleBinding

Grants **ClusterRole** to a **ServiceAccount**.



Vocabulary - Kubernetes

Auth Kubernetes

A plugin that connects Vault to Kubernetes. Holds **ServiceAccount** to **Policy** mappings

Policies

Provide a declarative way to grant or forbid access to certain paths and operations.

Vocabulary - Vault

Api-server



Verify service account

Login with
Service account

Pod



Pod



Expose UI



Login with
Username / Password

Our journey today

PRAQMA

Vault

```
# Exec into container
kubectl exec -it vault-2186619123-4xzk3 /bin/sh

# Set connection environment variable
$ export VAULT_ADDR=http://127.0.0.1:8200

# Init Vault
$ vault operator init
```

Output

```
# vault operator init
Key 1: m0kEP6N/M4pavLhgZWu86H0/R//FMQx825W...
Key 2: kuQlPA8bJH/KQQi7IbqAUra66h+iHyVTYT5...
Key 3: zCxQhafifnE2R7NcjH4T5MFwiMgMWc2xzDb...
Key 4: 000s3wVWLBpntjYSFR5jhQDePDBcFWsa7Dt...
Key 5: av93HNFBDvmzW4yoyScBV0LeV67LvKdjur0...

Initial Root Token:
012fccfd-1ed4-66b8-c030-36b2260d75c7

# Unseal Vault
$ vault operator unseal m0kEP6N/M4pavLh...
$ vault operator unseal zCxQhafifnE2R7N...
$ vault operator unseal av93HNFBDvmzW4y...
```

Initialize and unseal Vault

Vault

Create admin policy file

```
$ vi /vault/admin-policy.hcl
path "*" {
    capabilities = [ "create",
                    "read",
                    "update",
                    "list" ]
}
```

Create dev policy file

```
vi /vault/dev-policy.hcl
path "*" {
    capabilities = [ "read", "list" ]
}
```

Login as root

```
$ vault login 012fccfd-1ed4-66b8-c030-36...
```

Write policies to Vault

```
$ cd /vault
$ vault policy write admin admin-policy.hcl
$ vault policy write dev dev-policy.hcl
```

Prepare Vault policies

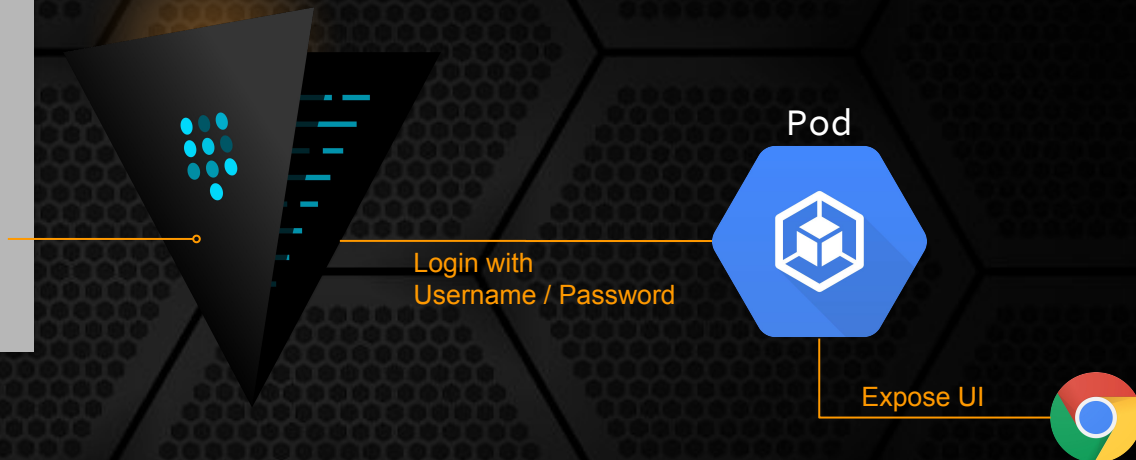
Vault

```
# Enable userpass authentication plugin
$ vault auth enable userpass

# Create an admin user
$ vault write auth/userpass/users/praqma \
  password="password" \
  policies="admin"

# Create an dev user
$ vault write auth/userpass/users/praqma-dev \
  password="password" \
  policies="dev"

# Test login from cli
$ vault login -method="userpass" \
  username="praqma"
```



Prepare Vault for UI

Service Account

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: vault-admin
```

Service
Account

Pod

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nwtool
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: nwtool
    spec:
      serviceAccountName: vault-admin
      containers:
        - name: nwtool
          image: pragma/network-multitool
          ports:
            - containerPort: 80
              name: nwtool
```

Token

Policy ↔ Service account mapping

```
vault write auth/kubernetes/role/admin
bound_service_account_names=vault-admin
bound_service_account_namespaces=default
policies=admin
ttl=10h
```

Validate

Service Account

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: vault-auth
```

TokenReview API



Create service accounts

PRAQMA

Vault

Exec into container

```
kubectl exec -it nwtool-982403051-qr96n bash
```

Find secret token

```
$ cd /var/run/secrets/kubernetes.io/serviceaccount  
$ cat token
```

Login with Curl

```
$ curl \  
  --request POST \  
  --data '{"jwt": "<TOKEN>", "role": "admin"}' \  
  http://vault:8200/v1/auth/kubernetes/login
```

Output

Output from Curl

```
{  
  ....  
  "auth": {  
    "client_token":  
      "c2029c0e-cca4-516e-aac3-59795e3...",  
    "accessor": "07e5fca8-32b0-dea3-c425-8...",  
    "policies": [  
      "default",  
      "admin"  
    ],  
    "metadata": {  
      "role": "admin",  
      "service_account_name": "vault-admin",  
      "service_account_namespace": "default",  
      "service_account_secret_name":  
        "vault-admin-token-xtt46",  
      ....  
    },  
    "lease_duration": 36000,  
    ....  
  }  
}
```

Get access from pod

Vault

```
# Put Vault token in variable
$ export CLIENT_TOKEN="c2029c0e-cca4-51..."

# Create new secret cloud
$ curl \
  --header "X-Vault-Token: $CLIENT_TOKEN" \
  --request POST \
  --data '{"cloud": "native"}' \
  http://vault:8200/v1/secret/foo

# Retrieve the secret with Curl
$ curl \
  --header "X-Vault-Token: $CLIENT_TOKEN" \
  http://vault:8200/v1/secret/foo
```

Output

```
# Output from Curl
{
  "data": {
    "cloud": "native"
  },
  "lease_duration": 2764800,
  "renewable": false,
  "request_id": "5e246671-ec05-6fc8-9f93-4fe..."
}
```

Test access from pod



Basic Vault commands

<https://www.vaultproject.io/docs/commands/index.html>

Policies

<https://www.vaultproject.io/docs/concepts/policies.html>

Tokens

<https://www.vaultproject.io/docs/concepts/tokens.html>

Seal/Unseal

<https://www.vaultproject.io/docs/concepts/seal.html>

Lease, renew and revoke

<https://www.vaultproject.io/docs/concepts/lease.html>

Want to know more?