Vault and OpenShift

Secure store of secrets

/ Who are we

Jörn Stenkamp
HashiCorp

Staff Solutions Engineer

Infrastructure as a Service

SDN / Overlay Networks

Openstack

Cloud Operating Model (IaC + Network + Security)

Tom Morelly
SVA

System Engineer

IaC

Golang

Kubernetes

Tim Schöllhammer
SVA

System Engineer

IaC

Ansible

OpenShift

/ Agenda

1 / Problem Statement

2 / Hashicorp Vault

3 / Hands On

Problem Statement

/ Is there a problem?

- Secrets in OpenShift
  - Secrets are base64 encoded in the projects.
  - They are not encrypted at rest.
  - Cluster admins can see all the secrets of all projects
- What if i need the same secret in multiple locations?
- What if i need to use secrets with other peoples?
- Integrating secrets over multiple stages or CI/CD pipelines?
- Key rotation!?!

HashiCorp Overview

Leading Cloud Infrastructure Automation

Our software stack enables the provisioning, securing, connecting, and running of apps and the infrastructure to support them.

We unlock the cloud operating model for every business and enable their digital transformation strategies to succeed.

Vagrant

Packer

Terraform

Vault

Boundary

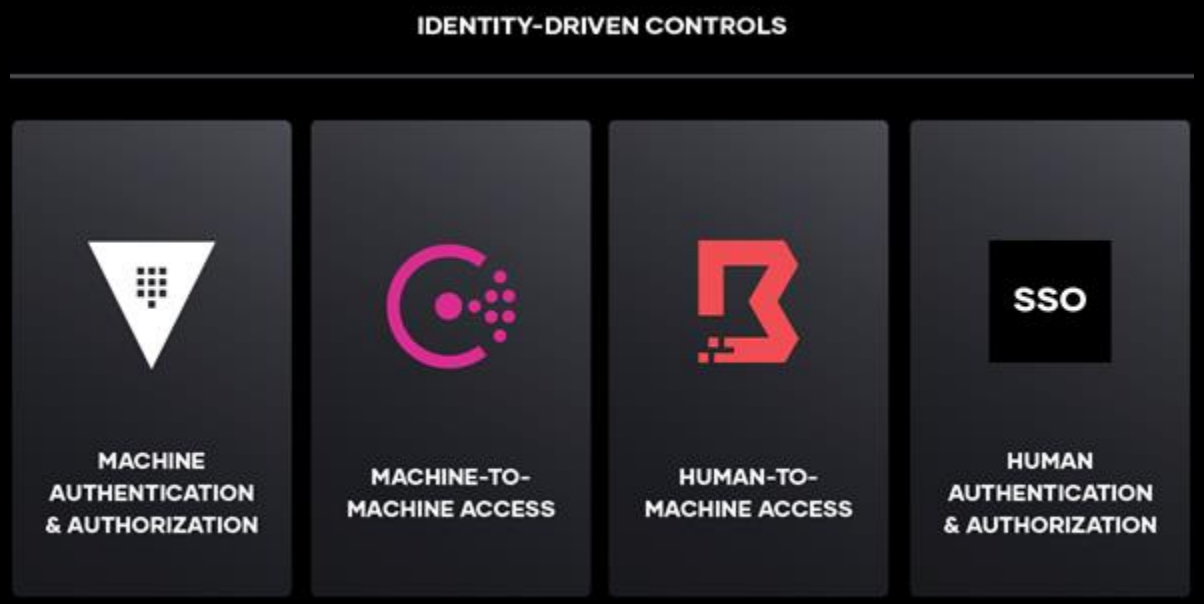
Consul

Nomad

Waypoint

ZERO TRUST, IDENTITY-BASED SECURITY

Trust Nothing. Authenticate and Authorize Everything.



https://www.hashicorp.com/resources/zero-trust-security-with-hashicorp-vault-consul-and-boundary

The 4 essential elements of distributed infrastructure

Connect

Infrastructure and applications

Development

Run applications

Security

Secure infrastructure and applications

Operations

Provision infrastructure

Hotel check-in process

How to get a Key-Card (Token) that grant you access to your room

1) You have to show your identity document (passport) and sign a document to verify your identity.

1) Once your identity is authenticated you get a key-card in return that contains a digital signature (a token) that belongs to your identity.

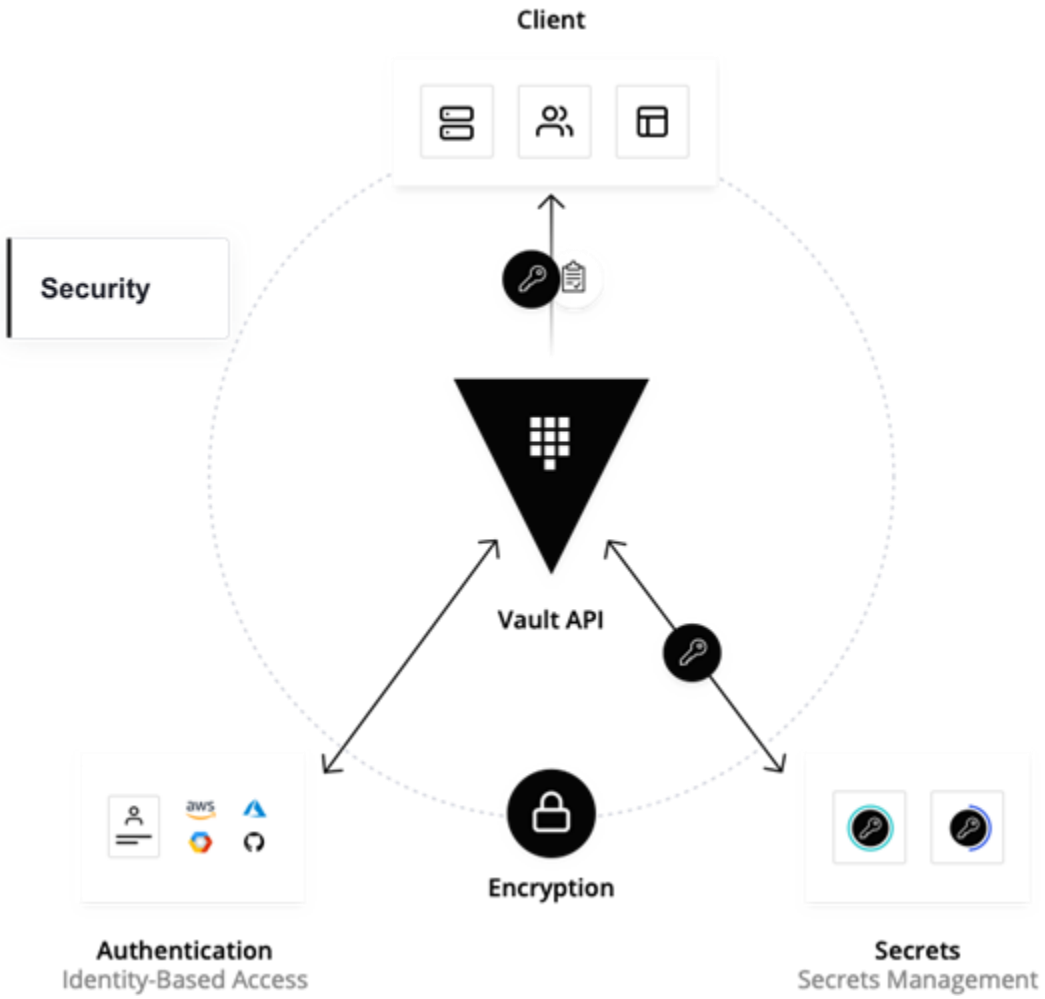
1) That key-card/token is authorized to open your room for the time of your stay.

Identity Access Management as a human-in-the-loop process.

Identity-based Security with Vault

Identity of requester authenticated against any identity model prior to granting access

Policies defined by the Security team and enforced at runtime.



Vault Principles

API Driven

Use policy to codify, protect, and automate access to secrets.

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    https://127.0.0.1:8200/v1/secret/config
```

Vault Principles

Secure with any Identity

Leverage any trusted identity provider, such as cloud IAM platforms, Kubernetes, Active Directory, to authenticate into Vault.

LDAP

U/P

...

Vault Principles

Extend and Integrate

Request secrets for any system through one consistent, audited, and secured workflow.

K/V

PKI

...

Vault Enterprise

Multi-Datacenter and Scale	Governance and Policy	Advanced Data Protection
• Replication • Replication Filters • Read Replicas • Path Filters	• Sentinel Integration • Control Groups • HSM Auto-unseal • Multi-factor Authentication • FIPS 140-2 & Seal Wrap • Entropy Augmentation	• KMIP • Transform (FPE, Data Masking)
Vault Enterprise Platform • Disaster Recovery • Namespaces		
Vault Open Source		

Vault Key Principles and Features

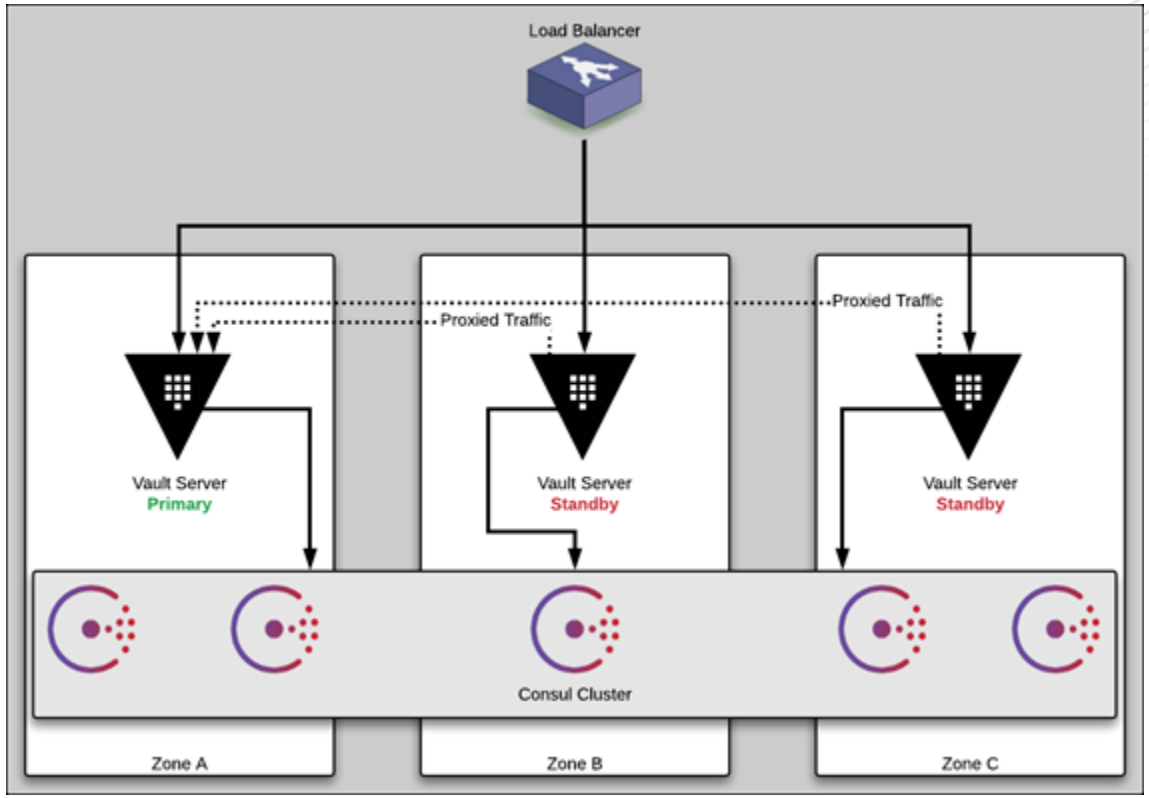
TTL and Lease

- Each authentication is attached to a token and it will be used for any subsequent requests. The token is configured with a TTL.
- The token can be revoked any time if needed or if it is compromised
- Dynamic secrets are attached to a lease that can be configured by roles. When lease expires, the secret is automatically deleted.

High Availability

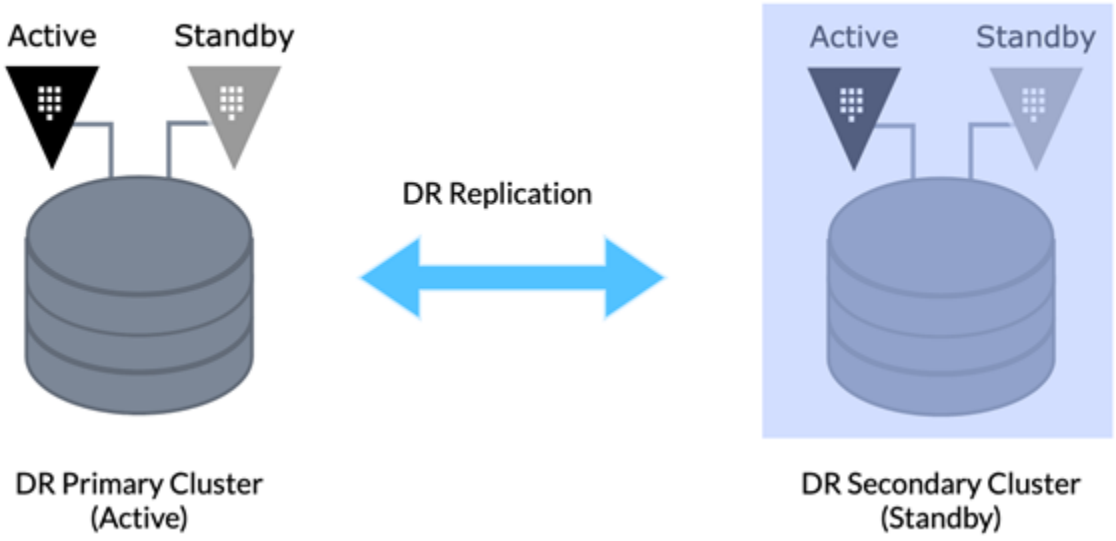
Vault Clustering

- Ensure High Availability at Cluster level
- A leader is elected, then other nodes are followers
- In case of the loss of the leader, another nodes will be elected as leader



Disaster Recovery

- All datas from primary cluster are replicated to secondary cluster
- In case of primary site loss, a promotion is done on the secondary site
- Applications can continue to work with minimum disruption



Vault Use Cases

Secrets as a Service

Managing access to secrets

- Secure your Static Secrets for already existing resources
- Leverage Dynamic Secrets to bring security to next level
- Combine ACL and Token lease to enforce security

PKI As A Service

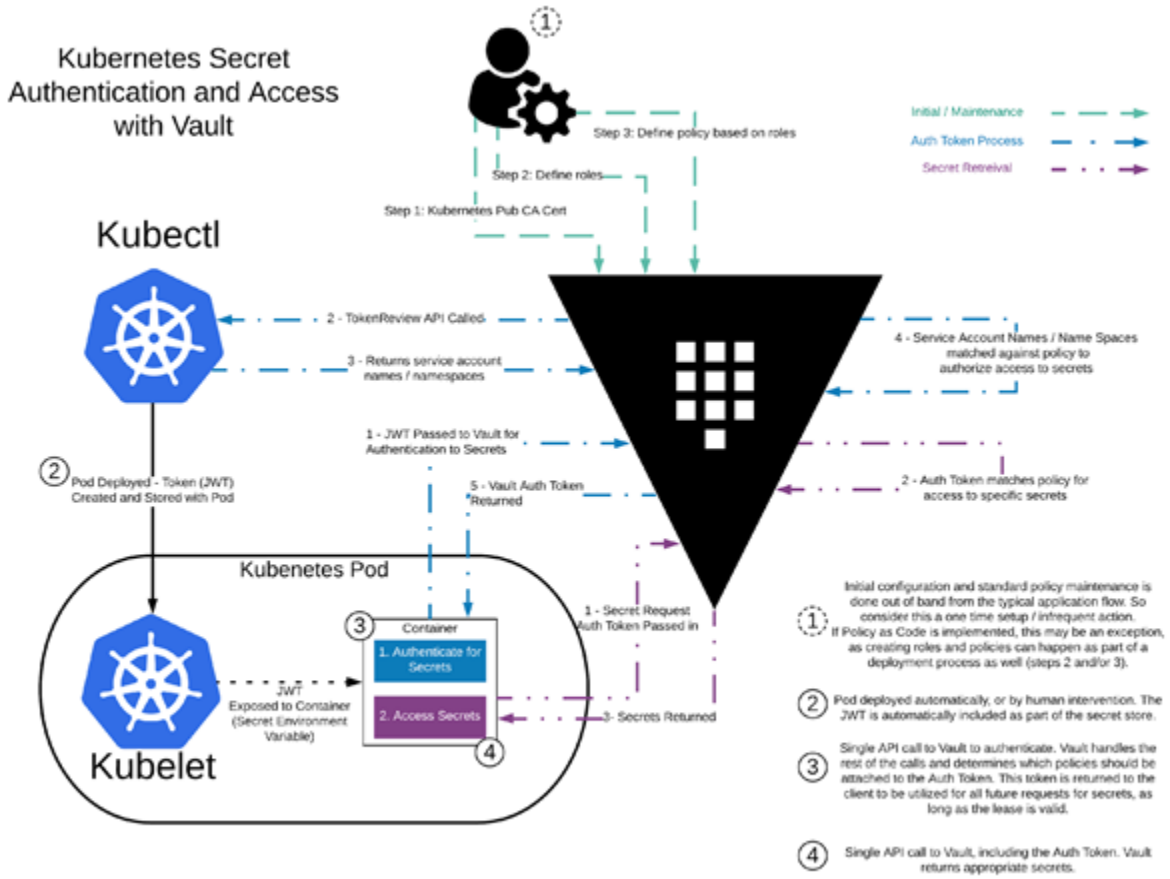
Delivering certificates programmatically

- Use Vault as an Intermediate Authority
- Automates your certificates generation
- Strengthen your security by rotating certificates more frequently

Securing your Container environment

Vault and Kubernetes integration

- Define Kubernetes as an Authentication Method
- Leverage service account and JWT Token to authenticates Apps
- Agent Sidecar Injector



Why Vault with Openshift ?

Improve secrets security in Openshift

Secrets are not stored in Openshift anymore

- **Leverage Kubernetes Authentication method to validate Pod's identity**
- **Retrieve static or dynamic secrets automatically**
- **More integration with Openshift to come**

Kubernetes Sidecar Secrets

Enable access to Vault secrets by Kubernetes applications that don't have native Vault logic built-in

Will allow **automatic injection of secrets into the pod file system** for static and dynamic secrets

Will allow **applications to only concern themselves with finding a secret at a filesystem** path, rather than managing the auth tokens and other mechanisms for direct interaction with Vault

Native Integration with Apps

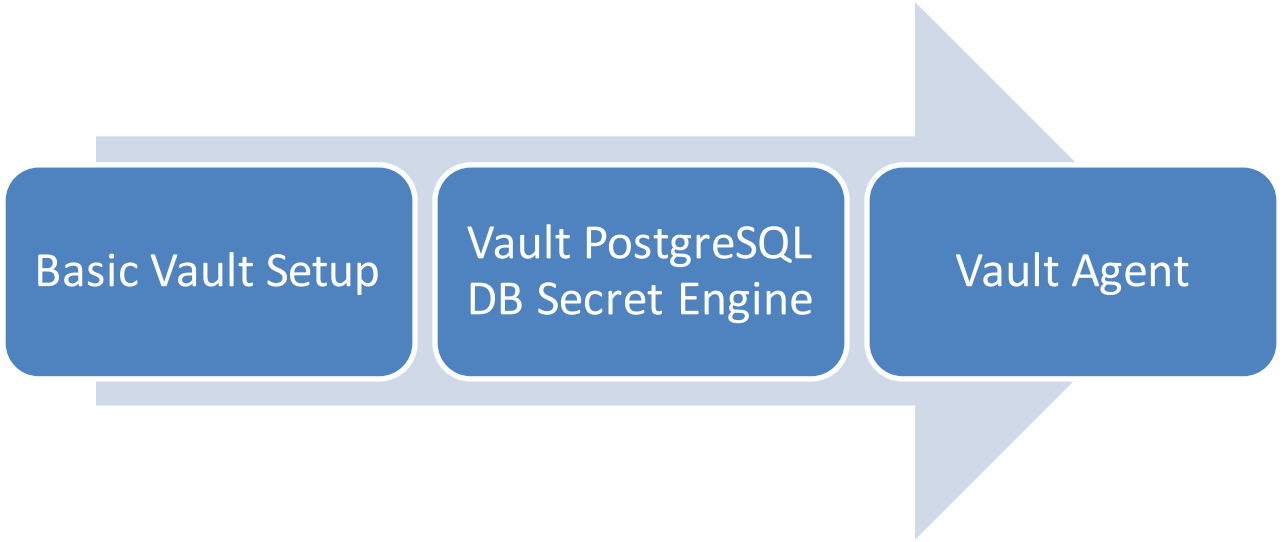
Enable access to Vault secrets by using native langage librairies and K8s authentication method

Hands on

/ Hands-On

Scenario:

Crediential Rotation using Vaults PostgreSQL Database Secret Engine in OpenShift



Goal:

Avoid hardcoded DB credientials; improve auditing

Basic Vault Setup

SVA

/ Installation

```
$> helm repo add hashicorp https://helm.releases.hashicorp.com
"hashicorp" has been added to your repositories
$> helm install vault hashicorp/vault --set "global.openshift=true" --set "server.dev.enabled=true"
```

https://github.com/hashicorp/vault-helm

/ Initialization

```
$> POD=$(oc get pods -lapp.kubernetes.io/name=vault --no-headers -o custom-columns=NAME:.metadata.name)
$> oc rsh $POD
$> [user@vault-demo-0] vault operator init --tls-skip-verify -key-shares=1 -key-threshold=1
```

/ Unsealing

```
$> [user@vault-demo-0] export KEYS=QzlUGvdPbIcM83UxyjuGd2ws7flZdNimQVCNbUvI2aU=
$> [user@vault-demo-0] export ROOT_TOKEN=s.UPBPfhDXYOtnv8mELhPA4br7
$> [user@vault-demo-0] export VAULT_TOKEN=$ROOT_TOKEN
$> [user@vault-demo-0] vault operator unseal --tls-skip-verify $KEYS
```

/ Configuration

```
$> [user@vault-demo-0] JWT=$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)
$> [user@vault-demo-0] KUBERNETES_HOST=https://${KUBERNETES_PORT_443_TCP_ADDR}:443
$> [user@vault-demo-0] vault auth enable --tls-skip-verify kubernetes
$> [user@vault-demo-0] vault write --tls-skip-verify auth/kubernetes/config token_reviewer_jwt=$JWT
kubernetes_host=$KUBERNETES_HOST kubernetes_ca_cert=@/var/run/secrets/kubernetes.io/serviceaccount
/ca.crt
```

https://www.vaultproject.io/docs/auth/kubernetes

Configuration

```
# allows listing and reading of secrets at path openshift/anwendertreffen
path "openshift/anwendertreffen" {
  capabilities = ["read", "list"]
}
```

```
$> vault policy write demo-policy demo-policy.hcl
$> vault write auth/kubernetes/role/demo-role \
    bound_service_account_names=default bound_service_account_namespaces='vault' \
    policies=demo-policy \
    ttl=2h
$> vault secrets enable -path=openshift kv
$> vault write openshift/anwendertreffen password=FooBar42!
```

/ Verify

openshift › anwendertreffen

anwendertreffen

JSON

```
1 {
2   "password": "FooBar42!"
3 }
```

```
$> vault kv get openshift/anwendertreffen
====== Data ======
Key         Value
---         -----
password    FooBar42!
```

Vault PostgreSQL DB Secret Engine

SVA

/ Setup

```
$> oc new-project psql
$> oc new-app postgresql-persistent \
    --name=postgresql -lname=postgresql  \
    --param DATABASE_SERVICE_NAME=postgresql --param POSTGRESQL_DATABASE=anwenderdb \
    --param POSTGRESQL_USER=user --param POSTGRESQL_PASSWORD=password \
    --param VOLUME_CAPACITY=1Gi \
    --env POSTGRESQL_ADMIN_PASSWORD=postgres
```

```
$> JWT=$(oc sa get-token default -n psql)
$> vault write auth/kubernetes/login role=demo-role jwt=${JWT}
$> VAULT_TOKEN=s.mCgDQH1SvtWT2lxdiqO2dvHj vault read openshift/anwendertreffen # from output before
Key                 Value
---                 -----
refresh_interval    768h
password            FooBar42!
```

/ Configuration

```
$> vault secrets enable database
$> vault write database/config/postgresql \
    plugin_name=postgresql-database-plugin \
    allowed_roles="psql-role" \
    connection_url="postgresql://{{username}}:{{password}}@postgresql.psql.svc:5432
/anwenderdb?sslmode=disable" \
    username="user" \
    password="password"
```

/ Configuration

```
# Allow a token to get a secret from the generic secret backend for the client role.
path "database/creds/psql-role" {
  capabilities = ["read"]
}

# allows listing and reading of secrets at path openshift/anwendertreffen
path "openshift/anwendertreffen" {
  capabilities = ["read", "list"]
}
```

```
$> vault policy write psql-policy psql-policy.hcl
$> vault write database/roles/psql-role \
    db_name=postgresql \
    creation_statements="CREATE ROLE \"\" WITH LOGIN PASSWORD '' VALID UNTIL ''; \
        GRANT SELECT ON ALL TABLES IN SCHEMA public TO \"\";" \
    default_ttl="1h" \
    max_ttl="24h"
$> vault write auth/kubernetes/role/demo-role \
    bound_service_account_names=default bound_service_account_namespaces='psql' \
    policies=psql-policy \
    ttl=2h
```

/ Verify

```
$>vault read database/creds/psql-role
Key                 Value
---                 -----
lease_id            database/creds/psql-role/SgASHJuiPt1zZ9YnEl8D9frA
lease_duration      1h
lease_renewable     true
password            A1a-PJrJpBGrTf4huKvs
username            v-root-psql-rol-M7dHciPlkyCOFYpEr8Vf-1625043951
```

Connecting to postgresql

```
sh-4.4$
sh-4.4$ psql
psql (10.15)
Type "help" for help.

postgres=# \du
                                                    List of roles
                    Role name                    |                         Attributes                         | Member of
-------------------------------------------------+------------------------------------------------------------+-----------
 postgres                                        | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
 user                                            | Create role                                                | {}
 v-root-psql-rol-DPkrXVCCB0hqykNuQI3x-1625043923 | Password valid until 2021-06-30 10:05:28+00                | {}
 v-root-psql-rol-M7dHciPlkyCOFYpEr8Vf-1625043951 | Password valid until 2021-06-30 10:05:56+00                | {}
 v-root-psql-rol-dvUIMOja5BteId49P9QK-1625043924 | Password valid until 2021-06-30 10:05:29+00                | {}

postgres=#
```

Lessons learned

/ Lessons Learned

- helm install vault ..

- vault operator init ..
- vault operator unseal ..

- vault auth enable kubernetes && vault write auth/kubernetes/config {JWT=$$}
- vault policy write psql-policy && vault write auth/kubernetes/role/psql-role {policies='psql-policy', ns='psql, vault'}

- vault secrets enable database && vault write database/config/postgresql {role=psql-role, psql-uri='...'}
- vault write database/roles/psql-role {db_name='postgresql', creation_statement='..'}

- vault read database/creds/psql-role

Vault Agent

/ Configuration

```
vault {
        tls_skip_verify = true
        address = "https://vault.apps.tld"
    }

    pid_file = "/var/run/secrets/vaultproject.io/pid"
    auto_auth {
            method "kubernetes"  {
                    type = "kubernetes"
                    mount_path = "auth/kubernetes"
                    config = {
                            role = "demo-role"
                            jwt = "@/var/run/secrets/kubernetes.io/serviceaccount/token"
                    }
            }
            sink "file" {
                    type = "file"
                    config = {
                            path = "/var/run/secrets/vaultproject.io/token"
                    }
            }
    }

    template {
        source      = "/vault/config/template.ctmpl"
        destination = "/var/run/secrets/vaultproject.io/application.properties"
    }
```

Resources

/ Resources

- https://falcosuessgott.github.io/openshift-vault-demo/
- https://www.openshift.com/blog/integrating-hashicorp-vault-in-openshift-4
- https://www.openshift.com/blog/managing-secrets-openshift-vault-integration
- https://www.openshift.com/blog/integrating-vault-with-legacy-applications
- https://www.vaultproject.io/docs/platform/k8s/helm/openshift
- https://www.vaultproject.io/docs/secrets/databases
- https://www.vaultproject.io/docs/secrets/databases/postgresql

Thanks

Why Vault with Openshift ?

Improve secrets security in Openshift

Secrets are not stored in Openshift anymore

- **Leverage Kubernetes Authentication method to validate Pod's identity**
- **Retrieve static or dynamic secrets automatically**
- **More integration with Openshift to come**

Kubernetes Sidecar Secrets

Enable access to Vault secrets by Kubernetes applications that don't have native Vault logic built-in

Will allow **automatic injection of secrets into the pod file system** for static and dynamic secrets

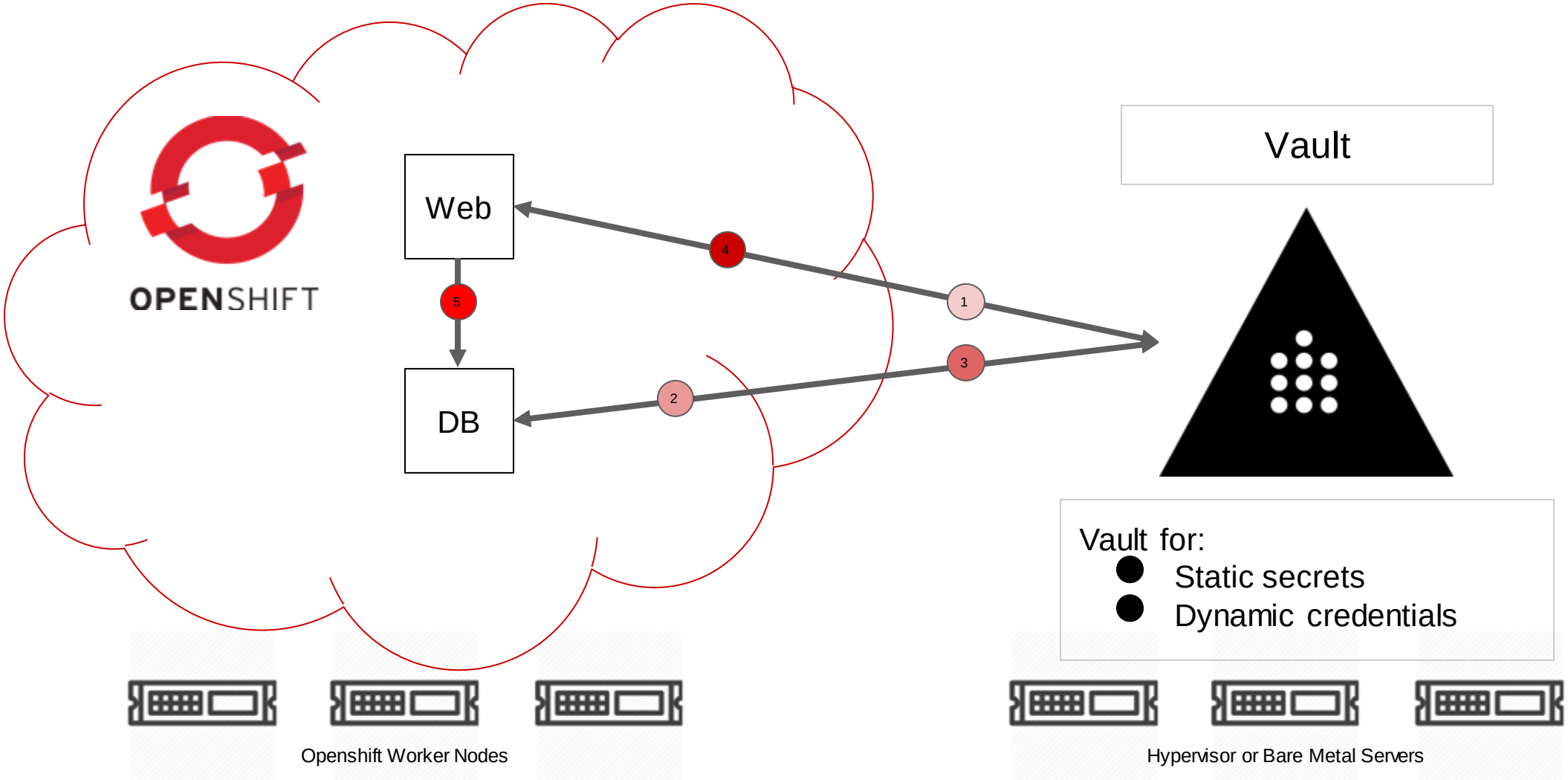
Will allow **applications to only concern themselves with finding a secret at a filesystem** path, rather than managing the auth tokens and other mechanisms for direct interaction with Vault

Native Integration with Apps

Enable access to Vault secrets by using native langage librairies and K8s authentication method

Vault and OpenShift Architecture

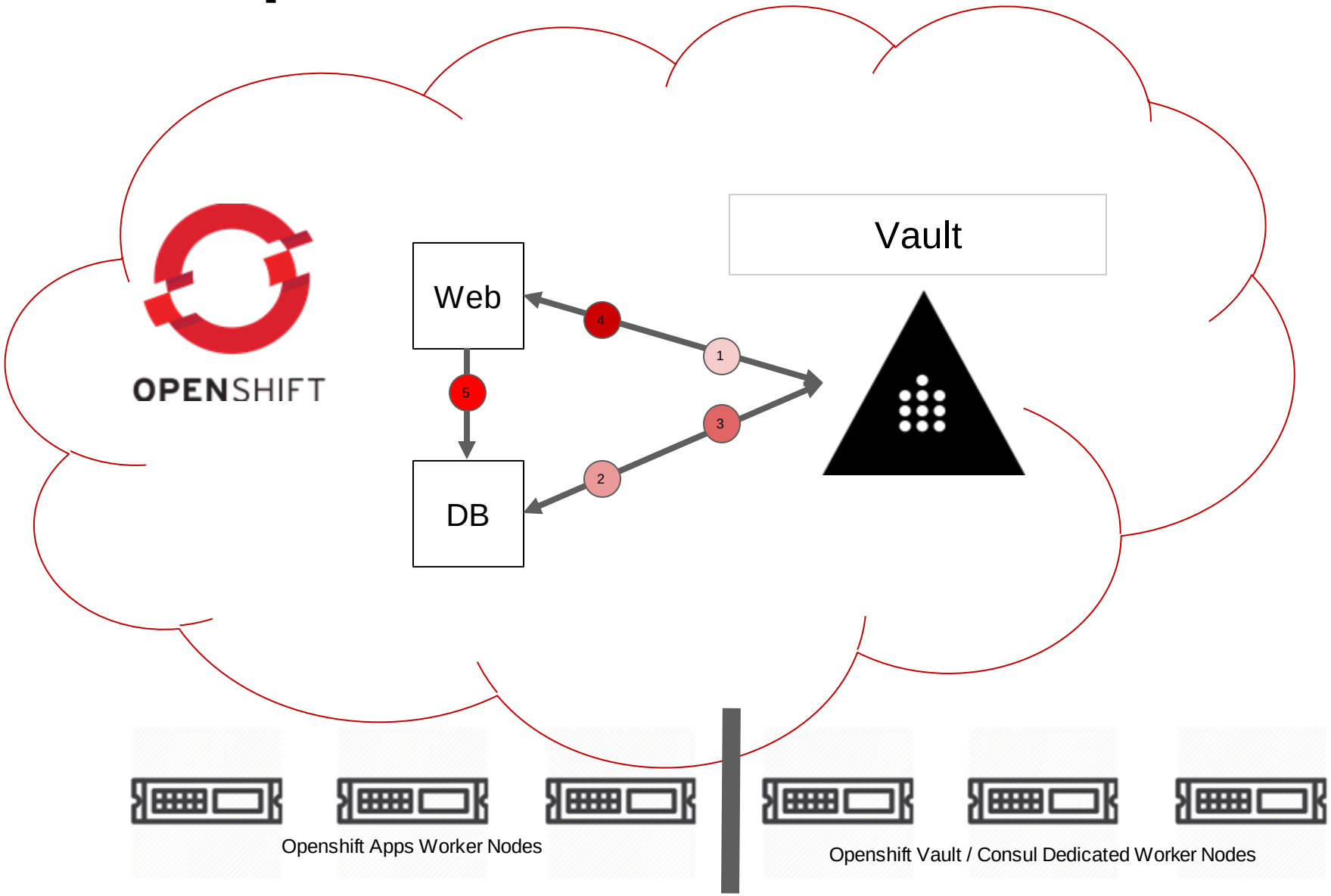
Vault Outside Openshift



Vault Outside Openshift

- Deploy Vault on existing Hypervisors Solution to isolate as much as possible the service from other processes
- Decouple Secrets management from Containers / PaaS platform
- Deliver secrets to legacy and containerized applications
- Leverage existing Load Balancer and Firewall Infrastructure
- Easy to hardened
- Need an automated process for lifecycle management like Configuration Management tools

Vault Inside Openshift



Vault Inside Openshift

- **Leverage Orchestrator features and Helm Chart for ease of deployment**
- **Close to Cloud Native Applications**
- **Access Vault from OpenShift Route for outside world**
- **Need more considerations regarding security aspects:**
  - Dedicated Worker Nodes
  - Cluster RBAC

Vault Kubernets Ref. Architecture: https://learn.hashicorp.com/vault/getting-started-k8s/k8s-reference-architecture

Vault kubernetes Security Considerations : https://learn.hashicorp.com/vault/getting-started-k8s/k8s-security-concerns

Thank you

hello@hashicorp.com

www.hashicorp.com