River's Notes

Dynamic secret with Vault in k8s

River Yang

2021-01-03 1019 words 5 minutes

CONTENTS

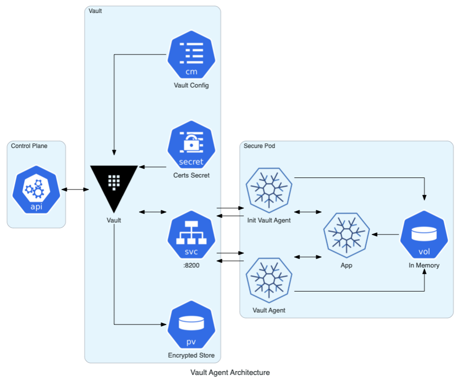
It's been a while since I updated last post... 😂

I promised to my interest group that will throw a demo for `Dynamic secret with Vault in k8s`, hmm, why don't just create a page for this?

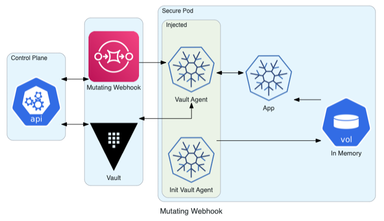
# Introduction for the things in the post

- `Vault` - https://www.vaultproject.io `TL;DR...` -> yeah, read yourself, I'm not writing repeating stuff...
- `k8s` - https://kubernetes.io `SAME AS ABOVE...`
- `jq` - https://stedolan.github.io/jq/ `ALSO SAME AS ABOVE...`
- `git` ...

# Overview



Vault Agent Architecture



# Prepare my workspace

For everyone's convenient, I'm going to use `minikube` today for demo: https://minikube.sigs.k8s.io/docs/start/

Here is how to provision my workspace (MAC OS):

Bash

```bash
# Install minikube (no virtualbox installed)
brew install minikube

# Start a k8s cluster
minikube start

# Verify the k8s cluster
kubectx # list current context, not important...
kubectl get all --all-namespaces

# Deploy a sample app and check how it goes...
kubectl create deployment --image nginx nginx

# Forward the port from local to k8s
kubectl port-forward $(kubectl get pods --output jsonpath='{.items[].metadata.name}') 8080:

# Clone the git repo...
git clone git@github.com:riveryc/aus-devops-group.git
```

# Create a Vault cluster in Kubernetes

What do we need:

- Vault Server running in k8s
- Unseal the vault
- Vault UI for visualization
- Vault injector
- Configure kubernetes auth in Vault

Bash

```
# Get vault deployed on Kubernetes
kubectl -n vault-example apply -f server/.

# Init Vault server with only one key only... -> Demo, DO NOT RUN THIS IN PROD!!
kubectl -n vault-example exec -it vault-example-0 -- vault operator init -key-shares=1 -key
VAULT_UNSEAL_KEY=$(cat cluster-keys.json | jq -r ".unseal_keys_b64[]")

# Check the status of vault
kubectl -n vault-example exec -it vault-example-0 -- vault status
# Unseal the vault
kubectl -n vault-example exec -it vault-example-0 -- vault operator unseal $VAULT_UNSEAL_KE

# Check the status of vault again
kubectl -n vault-example exec -it vault-example-0 -- vault status

# Check vault UI console if you want: https://localhost:8200/ & login with root token saved
kubectl -n vault-example port-forward vault-example-0 8200

# Deploy injector
kubectl -n vault-example apply -f injector/.

# Enable kubernetes auth in vault
TOKEN=$(cat cluster-keys.json | jq ".root_token" -r)
kubectl -n vault-example exec -it vault-example-0 -- vault login $TOKEN
kubectl -n vault-example exec -it vault-example-0 -- vault auth enable kubernetes
kubectl -n vault-example exec -it vault-example-0 -- sh

# In container:
vault write auth/kubernetes/config \
    token_reviewer_jwt="$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)" \
    kubernetes_host=https://${KUBERNETES_PORT_443_TCP_ADDR}:443 \
    kubernetes_ca_cert=@/var/run/secrets/kubernetes.io/serviceaccount/ca.crt
```

# Basic secret rotation:

Bash

```bash
# Create role for our app. The configuration below maps our Kubernetes service account, use
vault write auth/kubernetes/role/basic-secret-role \
    bound_service_account_names=basic-secret \
    bound_service_account_namespaces=vault-example \
    policies=basic-secret-policy \
    ttl=1h

# Create the policy to map our service account to a bunch of secrets.
cat <<EOF > /home/vault/app-policy.hcl
path "secret/basic-secret/*" {
  capabilities = ["read"]
}
EOF

vault policy write basic-secret-policy /home/vault/app-policy.hcl

# Create a kv secret, and make its ttl as 1m
vault secrets enable -path=secret/ kv

vault kv put secret/basic-secret/helloworld ttl=1m username=dbuser password=vErySecUr3P@ssw

#-------------------------------------------------------------------------------------------

# Create a workload pod to use this secret
kubectl -n vault-example apply -f example-apps/basic-secret/deployment.yaml

## Monitor the vault-agent container
kubectl -n vault-example logs -f $(kubectl -n vault-example get po -l "app=basic-secret" -o

# Check the secret inside of the pod
kubectl -n vault-example exec -it $(kubectl -n vault-example get po -l "app=basic-secret" -

# Change the secret value from UI, check the log of vault-agent, then refresh the secret fi
```

👉 **River's Notes**

## Database secret rotation:

Bash

```bash
# Deploy a postgres instance
kubectl create ns postgres
kubectl -n postgres apply -f example-apps/dynamic-postgresql/postgres.yaml
kubectl -n postgres apply -f example-apps/dynamic-postgresql/pgadmin.yaml

kubectl -n postgres exec -it $(kubectl -n postgres get pods -l "app=postgres" -o jsonpath="


# Enable database engine in vault
kubectl -n vault-example exec -it vault-example-0 -- vault secrets enable database

# Configure DB Credential creation
kubectl -n vault-example exec -it vault-example-0 -- sh

# In Container
vault write database/config/postgresdb \
    plugin_name=postgresql-database-plugin \
    allowed_roles="sql-role" \
    connection_url="postgresql://{{username}}:{{password}}@postgres.postgres:5432/postgresd
    username="postgresadmin" \
    password="admin123"

vault write database/roles/sql-role \
    db_name=postgresdb \
    creation_statements="CREATE ROLE \"{{name}}\" WITH LOGIN PASSWORD '{{password}}' VALID
        GRANT SELECT ON ALL TABLES IN SCHEMA public TO \"{{name}}\";" \
    default_ttl="1m" \
    max_ttl="2m"


# Test with vault, make sure the dynamic credential is valid ===> username = v-<UserName>-<
vault read database/creds/sql-role

#------------------------------------------------------------------------------------------
# Forward the port to access pgadmin: http://localhost:8080 in another cmd tab
kubectl -n postgres port-forward $(kubectl -n postgres get po -l "app=pgadmin" -o jsonpath=

#------------------------------------------------------------------------------------------

# Create policy for read postgres database

cat <<EOF > /home/vault/postgres-app-policy.hcl
path "database/creds/sql-role" {
  capabilities = ["read"]
}
EOF

vault policy write postgres-app-policy /home/vault/postgres-app-policy.hcl


# Allow Kubernetes to use service account get this role
vault write auth/kubernetes/role/sql-role \
    bound_service_account_names=dynamic-postgres \
    bound_service_account_namespaces=vault-example \
    policies=postgres-app-policy \
    ttl=1h


# Create a workload pod to start using this secret
kubectl -n vault-example apply -f example-apps/dynamic-postgresql/deployment.yaml

## Monitor the vault-agent container
kubectl -n vault-example logs -f $(kubectl -n vault-example get po -l "app=dynamic-postgres

# Verify the secrets in the pod, run it after 1-2 mins
kubectl -n vault-example exec -it $(kubectl -n vault-example get po -l "app=dynamic-postgre
```

Happy `vaulting` ~

R

Vault, k8s

Back | Home

Get You a Free SSL Cert

CONTENTS