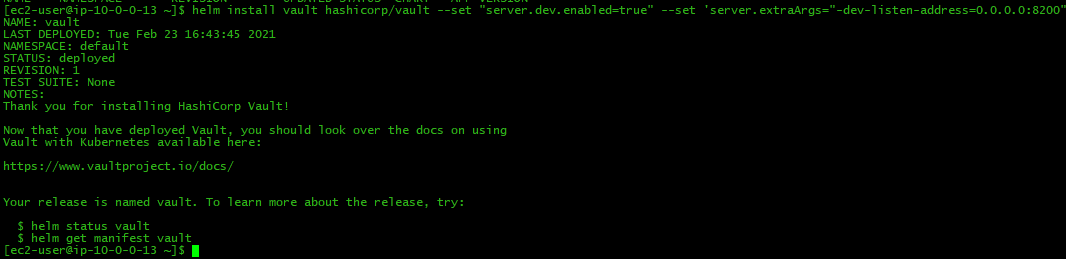
**HashiCorp Vault secret Injection with Side Car**

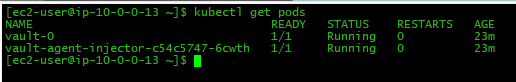
**Setup Vault Secret**

1. Install the Vault Helm chart

$ helm repo add hashicorp <https://helm.releases.hashicorp.com>

1. Install the latest version of the Vault server running in development mode.

$ helm install vault hashicorp/vault --set "server.dev.enabled=true" --set 'server.extraArgs="-dev-listen-address=0.0.0.0:8200"'

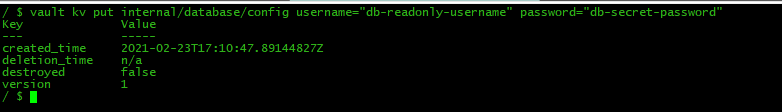
We could see Vault Pods are initialized.

1. Set a secret in Vault by “exec” into the vault pod and enable secret with path as “internal”

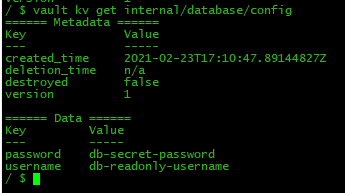
$ kubectl exec -it vault-0 -- /bin/sh

$ vault secrets enable -path=internal kv-v2

1. Create a KV with your data at path internal/database/config

$ vault kv put internal/database/config username="db-readonly-username" password="db-secret-password"

1. Verify that the secret is defined at the path internal/database/config and now the The secret is ready for the application.

 $ vault kv get internal/database/config

**Configure Kubernetes authentication.**

1. Enable the Kubernetes authentication method.

$ vault auth enable kubernetes

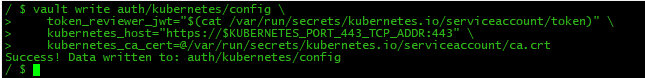
1. Configure the Kubernetes authentication method to use the service account token, the location of the Kubernetes host, and its certificate.

$ vault write auth/kubernetes/config \

token\_reviewer\_jwt="$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)" \

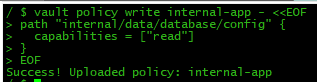
kubernetes\_host="https://$KUBERNETES\_PORT\_443\_TCP\_ADDR:443" \

[kubernetes\_ca\_cert=@/var/run/secrets/kubernetes.io/serviceaccount/ca.crt](mailto:kubernetes_ca_cert=@/var/run/secrets/kubernetes.io/serviceaccount/ca.crt)



1. For a client to read the secret data defined at internal/database/config, requires that the read capability be granted for the path internal/data/database/config. A policy defines a set of capabilities.

Write out the policy named internal-app that enables the read capability for secrets at path internal/data/database/config.



$ vault policy write internal-app - <<EOF

path "internal/data/database/config" {

capabilities = ["read"]

}

EOF

1. Create a Kubernetes authentication role named internal-app and exit the pod.

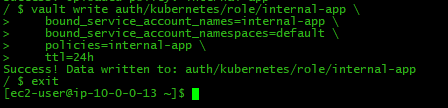
$ vault write auth/kubernetes/role/internal-app \

bound\_service\_account\_names=internal-app \

bound\_service\_account\_namespaces=default \

policies=internal-app \

ttl=24h



**Define a Kubernetes service account**

1. Kubernetes service account named internal-app

$ cat <<'EOF' >> service-account-internal-app.yml

apiVersion: v1

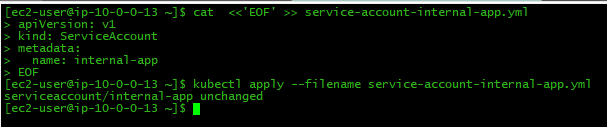
kind: ServiceAccount

metadata:

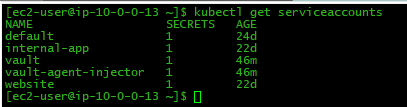
name: internal-app

EOF

and apply it

$ kubectl apply --filename service-account-internal-app.yml

1. Verify that the service account has been created

$ kubectl get serviceaccounts

1. Deploy the application and check if you’re able to access the password.

$ cat <<'EOF' >> deployment-orgchart.yml

apiVersion: apps/v1

kind: Deployment

metadata:

name: orgchart

labels:

app: orgchart

spec:

selector:

matchLabels:

app: orgchart

replicas: 1

template:

metadata:

annotations:

labels:

app: orgchart

spec:

serviceAccountName: internal-app

containers:

- name: orgchart

image: jweissig/app:0.0.1

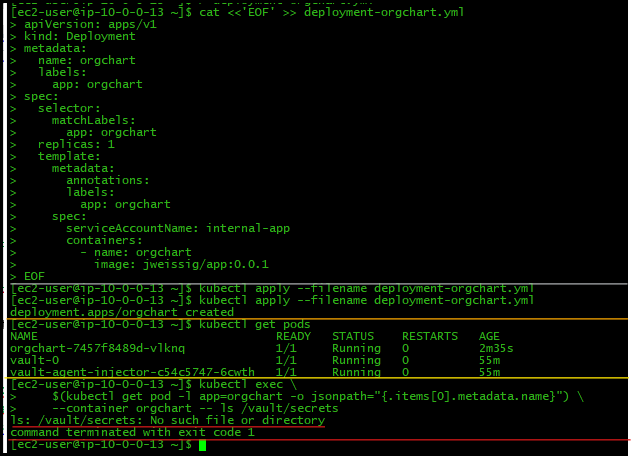
EOF

$ kubectl apply --filename deployment-orgchart.yml

1. Check the pods created by the deployment and try to exec into it and run command to list the secret  
     
   $ kubectl exec \

$(kubectl get pod -l app=orgchart -o jsonpath="{.items[0].metadata.name}") \

--container orgchart -- ls /vault/secrets



1. Inject secrets into the pod and once the deployment is patched, old pods will terminate and new pods with a init pod will initialize

$ cat <<'EOF' >> patch-inject-secrets.yml

spec:

template:

metadata:

annotations:

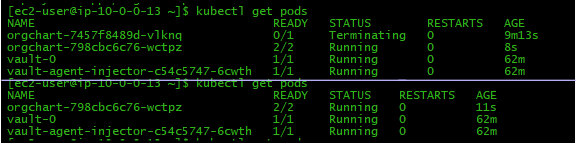
vault.hashicorp.com/agent-inject: "true"

vault.hashicorp.com/role: "internal-app"

vault.hashicorp.com/agent-inject-secret-database-config.txt: "internal/data/database/config"

EOF

$ kubectl patch deployment orgchart --patch "$(cat patch-inject-secrets.yml)"

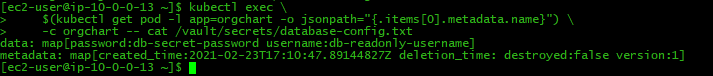


1. Now we can read the secrets from the init container by running below command

$ kubectl exec \

> $(kubectl get pod -l app=orgchart -o jsonpath="{.items[0].metadata.name}") \

> -c orgchart -- cat /vault/secrets/database-config.txt



Instead of patching the deployment, we can add the annotations in the Pod Spec in deployment file and apply it.

apiVersion: apps/v1

kind: Deployment

metadata:

name: orgchart

labels:

app: orgchart

spec:

template:

metadata:

annotations:

selector:

matchLabels:

app: orgchart

replicas: 1

template:

metadata:

annotations:

vault.hashicorp.com/agent-inject: "true"

vault.hashicorp.com/role: "internal-app"

vault.hashicorp.com/agent-inject-secret-database-config.txt: "internal/data/database/config"

labels:

app: orgchart

spec:

serviceAccountName: internal-app

containers:

- name: orgchart

image: jweissig/app:0.0.1