**Vault on Kubernetes Deployment tutorial**

Installing a single hashicorp vault cluster on a kubernetes cluster by using raft as integrated backend storage.

The [Vault Helm chart](https://github.com/hashicorp/vault-helm) is the recommended way to install and configure Vault on Kubernetes.

It is recommended to use namespace other than default to deploy applications in production for reasons like isolation, access management, upgrade management, and overall logical separation of default Kubernetes elements and overall aid in long-term management.

kubectl create namespace vault

**Setup helm Repo :**

helm repo add hashicorp https://helm.releases.hashicorp.com

"hashicorp" has been added to your repositories

The --dry-run flag will cause Helm to print the resulting YAML manifests that the Helm chart logically creates and applies.

To override default settings from the repo you can user override-vaules.yaml file. It will the subset. **override-values.yam**l file is providing a subset of values for attributes that are commonly overriden when deploying Vault to production on Kubernetes.

cat << EOF > ./override-values.yml

**#** Vault Helm Chart Value Overrides

global:

enabled: true

tlsDisable: false

injector:

enabled: true

**#** Use the Enterprise Image

image:

repository: "hashicorp/vault-enterprise"

tag: "1.5.0\_ent"

resources:

requests:

memory: 256Mi

cpu: 250m

limits:

memory: 256Mi

cpu: 250m

server:

**#** Use the Enterprise Image

image:

repository: "hashicorp/vault-enterprise"

tag: "1.5.0\_ent"

**#** These Resource Limits are in line with node requirements in the

**#** Vault Reference Architecture for a Small Cluster

resources:

requests:

memory: 8Gi

cpu: 2000m

limits:

memory: 16Gi

cpu: 2000m

**#** For HA configuration and because we need to manually init the vault,

**#** we need to define custom readiness/liveness Probe settings

readinessProbe:

enabled: true

path: "/v1/sys/health?standbyok=true&sealedcode=204&uninitcode=204"

livenessProbe:

enabled: true

path: "/v1/sys/health?standbyok=true"

initialDelaySeconds: 60

**#** extraEnvironmentVars is a list of extra enviroment variables to set with the stateful set. These could be

**#** used to include variables required for auto-unseal.

extraEnvironmentVars:

VAULT\_CACERT: /vault/userconfig/tls-ca/ca.crt

**#** extraVolumes is a list of extra volumes to mount. These will be exposed

**#** to Vault in the path `/vault/userconfig/<name>/`.

extraVolumes:

- type: secret

name: tls-server

- type: secret

name: tls-ca

- type: secret

name: kms-creds

**#** This configures the Vault Statefulset to create a PVC for audit logs.

**#** See https://www.vaultproject.io/docs/audit/index.html to know more

auditStorage:

enabled: true

standalone:

enabled: false

**#** Run Vault in "HA" mode.

ha:

enabled: true

replicas: 5

raft:

enabled: true

setNodeId: true

config: |

ui = true

listener "tcp" {

address = "[::]:8200"

cluster\_address = "[::]:8201"

tls\_cert\_file = "/vault/userconfig/tls-server/server.crt"

tls\_key\_file = "/vault/userconfig/tls-server/server.key"

tls\_ca\_cert\_file = "/vault/userconfig/tls-ca/ca.crt"

}

storage "raft" {

path = "/vault/data"

retry\_join {

leader\_api\_addr = "https://vault-0.vault-internal:8200"

leader\_ca\_cert\_file = "/vault/userconfig/tls-ca/ca.crt"

leader\_client\_cert\_file = "/vault/userconfig/tls-server/server.crt"

leader\_client\_key\_file = "/vault/userconfig/tls-server/server.key"

}

retry\_join {

leader\_api\_addr = "https://vault-1.vault-internal:8200"

leader\_ca\_cert\_file = "/vault/userconfig/tls-ca/ca.crt"

leader\_client\_cert\_file = "/vault/userconfig/tls-server/server.crt"

leader\_client\_key\_file = "/vault/userconfig/tls-server/server.key"

}

retry\_join {

leader\_api\_addr = "https://vault-2.vault-internal:8200"

leader\_ca\_cert\_file = "/vault/userconfig/tls-ca/ca.crt"

leader\_client\_cert\_file = "/vault/userconfig/tls-server/server.crt"

leader\_client\_key\_file = "/vault/userconfig/tls-server/server.key"

}

retry\_join {

leader\_api\_addr = "https://vault-3.vault-internal:8200"

leader\_ca\_cert\_file = "/vault/userconfig/tls-ca/ca.crt"

leader\_client\_cert\_file = "/vault/userconfig/tls-server/server.crt"

leader\_client\_key\_file = "/vault/userconfig/tls-server/server.key"

}

retry\_join {

leader\_api\_addr = "https://vault-4.vault-internal:8200"

leader\_ca\_cert\_file = "/vault/userconfig/tls-ca/ca.crt"

leader\_client\_cert\_file = "/vault/userconfig/tls-server/server.crt"

leader\_client\_key\_file = "/vault/userconfig/tls-server/server.key"

}

}

service\_registration "kubernetes" {}

**#** Vault UI

ui:

enabled: true

serviceType: "LoadBalancer"

serviceNodePort: null

externalPort: 8200

**#** For Added Security, edit the below

**#**loadBalancerSourceRanges:

**#** - < Your IP RANGE Ex. 10.0.0.0/16 >

**#** - < YOUR SINGLE IP Ex. 1.78.23.3/32 >

EOF

**Pod Resource Limits:** It is best practice to mention resource limits for pods. In the above override-values.yaml file, request limits are defined for the Vault Agent Injector and Vault server Pod(s) based on the requirements

server:

resources:

requests:

memory: 8Gi

cpu: 2000m

limits:

memory: 16Gi

cpu: 2000m

**Stateful Storage:** Both the [dataStorage](https://www.vaultproject.io/docs/platform/k8s/helm/configuration#datastorage) and the [auditStorage](https://www.vaultproject.io/docs/platform/k8s/helm/configuration#auditstorage) stanzas of the Vault Helm chart values file make use of [PhysicalVolumeClaims](https://kubernetes.io/docs/concepts/storage/persistent-volumes) (PVC). This configures the Vault Statefulset to create a PVC for data storage when using the file or Raft storage backend.

server:

dataStorage:

enabled: **true**

auditStorage:

enabled: **true**

**Vault Seal Config:**

The seal portion of the Vault configuration specifies the seal type to use for additional data protection such as using hardware security module (HSM) or Cloud Key Management Server (KMS) solutions to encrypt and decrypt the Vault master key to automatically unseal Vault.

This is optional if this is not configured then you can unseal vault manually using the shamir algorithm.

seal "gcpckms" {

project = "<NAME OF PROJECT>"

region = "global"

key\_ring = "<NAME OF KEYRING>"

crypto\_key = "<NAME OF KEY>"

}

**Auto-Unseal :** you can auto unseal the vault with different options.

1. GCPKMS
2. AWS EKS AUTO UNSEAL

**GCP KMS:** Vault Helm requires the Google Cloud KMS credentials stored in credentials.json and mounted as a secret in each Vault server pod.

##### **Create the Secret - GCP KMS**

Before adding the seal to your configuration, create the secret in Kubernetes:

kubectl create secret generic kms-creds --from-file=credentials.json

Vault Helm mounts this to /vault/userconfig/kms-creds/credentials.json.

**Amazon EKS auto-unseal:**

Vault Helm requires the AWS credentials stored as environment variables that are defined in each Vault server pod.

##### **Create the Secret - AWS EKS**

Before adding the seal to your configuration, create a secret with your EKS access key/secret:

kubectl create secret generic eks-creds \

--from-literal=AWS\_ACCESS\_KEY\_ID="**$**{AWS\_ACCESS\_KEY\_ID?}" \

--from-literal=AWS\_SECRET\_ACCESS\_KEY="${AWS\_SECRET\_ACCESS\_KEY?}"

**Seal stanza:**

seal "awskms" {

region = "KMS\_REGION\_HERE"

kms\_key\_id = "KMS\_KEY\_ID\_HERE"

}

**Vault Storage Config:**

The storage portion of the HCL configures the storage backend, which represents the path location for the durable storage of Vault's data.

Below, retry\_join statements are used and reflect the number of replicas defined in the Vault Helm chart.

storage "raft" {

path = "/vault/data"

retry\_join {

leader\_api\_addr = "https://vault-0.vault-internal:8200"

leader\_ca\_cert\_file = "/vault/userconfig/vault-ca/ca.crt"

leader\_client\_cert\_file = "/vault/userconfig/vault-cert/tls.crt"

leader\_client\_key\_file = "/vault/userconfig/vault-key/tls.key"

}

retry\_join {

leader\_api\_addr = "https://vault-1.vault-internal:8200"

leader\_ca\_cert\_file = "/vault/userconfig/vault-ca/ca.crt"

leader\_client\_cert\_file = "/vault/userconfig/vault-cert/tls.crt"

leader\_client\_key\_file = "/vault/userconfig/vault-key/tls.key"

}

retry\_join {

leader\_api\_addr = "https://vault-2.vault-internal:8200"

leader\_ca\_cert\_file = "/vault/userconfig/vault-ca/ca.crt"

leader\_client\_cert\_file = "/vault/userconfig/vault-cert/tls.crt"

leader\_client\_key\_file = "/vault/userconfig/vault-key/tls.key"

}

}

Instead of joining manually using vault operator join we can use this retry-join.

**Vault service registration:**

Kubernetes Service Registration tags Vault pods with their current status for use with selectors. Service registration is only available when Vault is running in High Availability mode.

service\_registration "kubernetes" {}

you can also mention your namespace and pod\_name as follows:

service\_registration "kubernetes" {

namespace = "my-namespace"

pod\_name = "my-pod-name"

}

**Load balancers and replication:**

When using a load balancer in front of Vault, replication traffic should always be directed to the active Vault. When using the Helm chart and enabling the **server.service** attribute, you can choose from an internal "headless" ClusterIP, NodePort, or LoadBalancer.

**Helm chart annotation for nginx:**

kubernetes.io/ingress.class: nginx

kubernetes.io/tls-acme: 'true'

**Vault ui:**

To have a graphical user interface for vault we can use vault ui.

#### **Viewing the Vault UI**

If for security reasons you chose not to expose the UI external to the Kubernetes cluster, the Vault UI can also be exposed via port-forwarding.

In this case, you can expose the Vault UI with port-forwarding:

kubectl port-forward vault-0 8200:8200

Forwarding from 127.0.0.1:8200 -> 8200

Forwarding from [::1]:8200 -> 8200

#### **Liveness/Readiness Probes**

Probes are essential for detecting failures, rescheduling and using pods in Kubernetes. The Helm chart offers configurable readiness and liveliness probes which can be customized for a variety of use cases.

Vault's [/sys/health](https://learn.hashicorp.com/api-docs/system/health) endpoint can be customized to change the behavior of the health check. For example, we can change the Vault readiness probe to show the Vault pods are ready even if they're still uninitialized and sealed using the following probe:

server:

readinessProbe:

enabled: **true**

path: '/v1/sys/health?standbyok=true&sealedcode=204&uninitcode=204'

**Install vault**

helm install vault hashicorp/vault --namespace vault -f override-values.yaml

**Initialize and unseal vault**

See the pods that are running,

kubectl get pods -l app.kubernetes.io/name=vault -n vault

NAME READY STATUS RESTARTS AGE

vault-0 0/1 Running 0 1m49s

vault-1 0/1 Running 0 1m49s

vault-2 0/1 Running 0 1m49s

**initialize vault:**

kubectl exec -ti vault-0 -- vault operator init

unseal you vault using the unseal keys that you get from the above output

Repeat the unseal process for all Vault server pods. When all Vault server pods are unsealed they report READY 1/1.

**$** kubectl get pods -l app.kubernetes.io/name=vault

NAME READY STATUS RESTARTS AGE

vault-0 1/1 Running 0 1m49s

vault-1 1/1 Running 0 1m49s

vault-2 1/1 Running 0 1m49s

DIFFERENCES IN THE CONFIGURATIONS (override-values.yaml)

**Our config file that we used for vault deployment on eks–**

**override-vaules.yaml:**

server:

ui:

enabled: true

ha:

enabled: true

replicas: 3

raft:

enabled: true

service:

type: NodePort

port: 8200

targetPort: 8200

protocol: TCP

Remaining configurations will be default ones as shown in the following table.

**DIFFERENCES BETWEEN CONFIGURATIONS :**

|  |  |
| --- | --- |
| **DEFAULT CONFIGURATION THAT WE USED** | **CONFIGURATION FROM LINK** |
| **global:**  enabled: true  tlsDisable: true | **global:**  enabled: true  tlsDisable: false |
| **injector:**  enabled: true   |  | | --- | |  | | image:  repository:"hashicorp/vault-k8s"  tag: "0.6.0"  pullPolicy: IfNotPresent | |  |  | |  |  | |  |  | |  |  | | **injector:**   |  | | --- | |  |   enabled: true  # Use the Enterprise Image  image:  repository: "hashicorp/vault-enterprise"  tag: "1.5.0\_ent"  resources:  requests:  memory: 256Mi  cpu: 250m  limits:  memory: 256Mi  cpu: 250m |
| **server:**  **image**:  repository: "vault"  tag: "1.5.4"  # Overrides the default Image Pull Policy  pullPolicy: IfNotPresent  **readinessProbe:**  enabled: true  # If you need to use a http path instead of the default exec  # path: /v1/sys/health?standbyok=true  **livenessProbe**:  enabled: false  path: "/v1/sys/health?standbyok=true"  # When a probe fails, Kubernetes will try failureThreshold times before giving up  failureThreshold: 2  # Number of seconds after the container has started before probe initiates  initialDelaySeconds: 60  **extraEnvironmentVars**: {}  **extraVolumes**: []  **dataStorage**:  enabled: true  **auditStorage**:  enabled: false  **service**:  type: NodePort  port: 8200  targetPort: 8200  protocol: TCP  **standalone**:  enabled: false  **ha**:  enabled: true  replicas: 3  **raft**:  enabled: true  config: |  ui = true  listener "tcp" {  tls\_disable = 1  address = "[::]:8200"  cluster\_address = "[::]:8201"  }  storage "raft" {  path = "/vault/data"  }  service\_registration "kubernetes" {} | **server:**  # Use the Enterprise Image  **image**:  repository: "hashicorp/vault-enterprise"  tag: "1.5.0\_ent"  # These Resource Limits are in line with node requirements in the  # Vault Reference Architecture for a Small Cluster  **resources:**  requests:  memory: 8Gi  cpu: 2000m  limits:  memory: 16Gi  cpu: 2000m  **readinessProbe:**  enabled: true  path: "/v1/sys/health?standbyok=true&sealedcode=204&uninitcode=204"  **livenessProbe**:  enabled: true  path: "/v1/sys/health?standbyok=true"  initialDelaySeconds: 60  **extraEnvironmentVars**:  VAULT\_CACERT: /vault/userconfig/tls-ca/ca.crt  # extraVolumes is a list of extra volumes to mount. These will be exposed  # to Vault in the path `/vault/userconfig/<name>/`.  **extraVolumes:**  - type: secret  name: tls-server  - type: secret  name: tls-ca  - type: secret  name: kms-creds  # This configures the Vault Statefulset to create a PVC for audit logs.  # See https://www.vaultproject.io/docs/audit/index.html to know more  **auditStorage**:  enabled: true  **standalone**:  enabled: false  **ha**:  enabled: true  replicas: 5  **raft**:  enabled: true  setNodeId: true  config: |  ui = true  listener "tcp" {  address = "[::]:8200"  cluster\_address = "[::]:8201"  tls\_cert\_file = "/vault/userconfig/tls-server/server.crt"  tls\_key\_file = "/vault/userconfig/tls-server/server.key"  tls\_ca\_cert\_file = "/vault/userconfig/tls-ca/ca.crt"  }  storage "raft" {  path = "/vault/data"  retry\_join {  leader\_api\_addr = "https://vault-0.vault-internal:8200"  leader\_ca\_cert\_file = "/vault/userconfig/tls-ca/ca.crt"  leader\_client\_cert\_file = "/vault/userconfig/tls-server/server.crt"  leader\_client\_key\_file = "/vault/userconfig/tls-server/server.key"  }  retry\_join {  leader\_api\_addr = "https://vault-1.vault-internal:8200"  leader\_ca\_cert\_file = "/vault/userconfig/tls-ca/ca.crt"  leader\_client\_cert\_file = "/vault/userconfig/tls-server/server.crt"  leader\_client\_key\_file = "/vault/userconfig/tls-server/server.key"  }  retry\_join {  leader\_api\_addr = "https://vault-2.vault-internal:8200"  leader\_ca\_cert\_file = "/vault/userconfig/tls-ca/ca.crt"  leader\_client\_cert\_file = "/vault/userconfig/tls-server/server.crt"  leader\_client\_key\_file = "/vault/userconfig/tls-server/server.key"  }  retry\_join {  leader\_api\_addr = "https://vault-3.vault-internal:8200"  leader\_ca\_cert\_file = "/vault/userconfig/tls-ca/ca.crt"  leader\_client\_cert\_file = "/vault/userconfig/tls-server/server.crt"  leader\_client\_key\_file = "/vault/userconfig/tls-server/server.key"  }  retry\_join {  leader\_api\_addr = "https://vault-4.vault-internal:8200"  leader\_ca\_cert\_file = "/vault/userconfig/tls-ca/ca.crt"  leader\_client\_cert\_file = "/vault/userconfig/tls-server/server.crt"  leader\_client\_key\_file = "/vault/userconfig/tls-server/server.key"  }  }  service\_registration "kubernetes" {} |
| **ui**:  enabled: true  serviceType: "ClusterIP"  serviceNodePort: null  externalPort: 8200  # loadBalancerSourceRanges:  # - 10.0.0.0/16  # - 1.78.23.3/32 | **ui**:  enabled: true  serviceType: "LoadBalancer"  serviceNodePort: null  externalPort: 8200  # For Added Security, edit the below  #loadBalancerSourceRanges:  # - < Your IP RANGE Ex. 10.0.0.0/16 >  # - < YOUR SINGLE IP Ex. 1.78.23.3/32 > |

**Reference Links:**

<https://learn.hashicorp.com/tutorials/vault/kubernetes-raft-deployment-guide>

<https://github.com/hashicorp/vault-helm/blob/master/values.yaml>