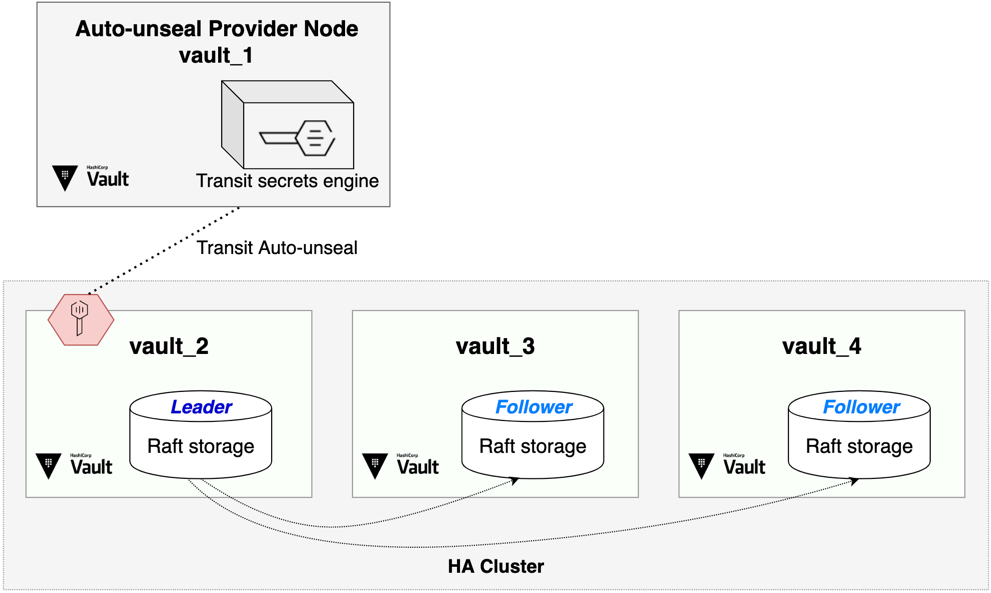
**VAULT HA MODE CONFIGURATION WITH RAFT BACKEND**



**vault\_1** is initialized and unsealed. The root token creates a transit key that enables the other Vaults auto-unseal. This Vault does not join the cluster.

**vault\_2** is initialized and unsealed. This Vault starts as the cluster leader. An example K/V-V2 secret is created.

**vault\_3** is only started. You will join it to the cluster.

**vault\_4** is only started. You will join it to the cluster.

**Prerequisites :**

* Install terraform

**Vault transit** : **auto-unseal a Vault with Transit secrets engine**

It is the encryption service provider, and its transit key protects the **Vault 2** server's master key, which is a part of HA cluster.

**PROCEDURE :**

git clone https://github.com/hashicorp/vault-guides.git

cd vault-guides/operations/raft-storage/aws

**$** export AWS\_ACCESS\_KEY\_ID = "<YOUR\_AWS\_ACCESS\_KEY\_ID>"

**$** export AWS\_SECRET\_ACCESS\_KEY = "<YOUR\_AWS\_SECRET\_ACCESS\_KEY>"

cp terraform.tfvars.example terraform.tfvars

Edit the tfvars file : give the region, availability zone ,key-pair in that region.

Initialize the terraform : terraform init

Now run the terraform : terraform apply

**OUTPUT :**

vault\_1 (54.145.131.93) | internal: (10.0.101.21)  
    - Initialized and unsealed.  
    - The root token creates a transit key that enables the other Vaults to auto-unseal.  
    - Does not join the High-Availability (HA) cluster.

  vault\_2 (18.205.103.30) | internal: (10.0.101.22)  
    - Initialized and unsealed.  
    - The root token and recovery key is stored in /tmp/key.json.  
    - K/V-V2 secret engine enabled and secret stored.  
    - Leader of HA cluster

    $ ssh -l ubuntu 18.205.103.30 -i vault-eks.pem

    # Root token:  
    $ ssh -l ubuntu 18.205.103.30 -i vault-eks.pem "cat ~/root\_token"  
    # Recovery key:  
    $ ssh -l ubuntu 18.205.103.30 -i vault-eks.pem "cat ~/recovery\_key"

  vault\_3 (34.238.120.59) | internal: (10.0.101.23)  
    - Started  
    - You will join it to cluster started by vault\_2

    $ ssh -l ubuntu 34.238.120.59 -i vault-eks.pem

  vault\_4 (3.91.145.125) | internal: (10.0.101.24)  
    - Started  
    - You will join it to cluster started by vault\_2

    $ ssh -l ubuntu 3.91.145.125 -i vault-eks.pem  
root@ip-172-31-19-186:~/vault-guides/operations/raft-storage/aws# vault status

Currently, **vault\_2** is initialized, unsealed, and has HA enabled. It is the only node in a cluster. The remaining nodes, **vault\_3** and **vault\_4**, have not joined its cluster.

ssh -l ubuntu 18.205.103.30 -i vault-eks.pem

sudo cat /etc/vault.d/vault.hcl

storage "raft" {

path = "/vault/vault\_2"

node\_id = "vault\_2"

}

listener "tcp" {

address = "0.0.0.0:8200"

cluster\_address = "0.0.0.0:8201"

tls\_disable = true

}

seal "transit" {

address = "http://10.0.101.21:8200"

token = "root"

disable\_renewal = "false"

// Key configuration

key\_name = "unseal\_key"

mount\_path = "transit/"

}

api\_addr = "http://18.205.103.30:8200"

cluster\_addr = "http://10.0.101.22:8201"

disable\_mlock = true

ui=true

Configure the vault CLI to use the root token for requests.

export VAULT\_TOKEN=$(cat ~/root\_token)

vault operator raft list-peers

using above command you can view the raft configuration.

**Add nodes to the cluster :**

**Vault-3 :**

ssh -l ubuntu 34.238.120.59 -i vault-eks.pem

**join vault-3 to vault-2 :**

**configuration file :**

storage "raft" {  
  path    = "/vault/vault\_3"  
  node\_id = "vault\_3"  
}

listener "tcp" {  
  address     = "0.0.0.0:8200"  
  cluster\_address     = "0.0.0.0:8201"  
  tls\_disable = true  
}

seal "transit" {  
  address            = "http://10.0.101.21:8200"  
  token              = "root"  
  disable\_renewal    = "false"

  // Key configuration  
  key\_name           = "unseal\_key"  
  mount\_path         = "transit/"  
}

api\_addr = "http://34.238.120.59:8200"  
cluster\_addr = "http://10.0.101.23:8201"  
disable\_mlock = true  
ui=true

vault operator raft join http://18.205.103.30 :8200

Key Value

--- -----

Joined true

Configure the vault CLI to use **vault\_2** root token for requests which is stored in the ~/root\_token file on the **vault\_2** host.

export VAULT\_TOKEN="s.qr20f2k5rtQ08UjmdaxkdByV"

Examine the raft peer set using ,

vault operator raft list-peers

at this point you will be able to see vault-2 and vault-3 in the list.

vault kv get kv/apikey

using the above command you will be able to see the secrets in vault-2 from

vault-3

**Join vault-4 :**

ssh -l ubuntu 3.91.145.125 -i vault-eks.pem

sudo vi /etc/vault.d/vault.hcl

sudo systemctl stop vault

As we know vault-2 and vault-3 ips we can add them to retry-join .Then start the vault.

storage "raft" {  
  path    = "/vault/vault\_4"  
  node\_id = "vault\_4"  
retry\_join {  
    leader\_api\_addr = "http://18.205.103.30:8200"  
  }  
  retry\_join {  
    leader\_api\_addr = "http://34.238.120.59:8200"  
  }  
}

listener "tcp" {  
  address     = "0.0.0.0:8200"  
  cluster\_address     = "0.0.0.0:8201"  
  tls\_disable = true  
}

seal "transit" {  
  address            = "http://10.0.101.21:8200"  
  token              = "root"  
  disable\_renewal    = "false"

  // Key configuration  
  key\_name           = "unseal\_key"  
  mount\_path         = "transit/"  
}

api\_addr = "http://3.91.145.125:8200"  
cluster\_addr = "http://10.0.101.24:8201"  
disable\_mlock = true  
ui=true

sudo systemctl start vault

**configure your vault :**

export VAULT\_TOKEN="s.qr20f2k5rtQ08UjmdaxkdByV"

**List you vault peers :**

vault operator raft list-peers  
Node       Address             State       Voter  
----       -------             -----       -----  
vault\_2    10.0.101.22:8201    leader      true  
vault\_3    10.0.101.23:8201    follower    true  
vault\_4    10.0.101.24:8201    follower    true

**URL’s to connect to the vault-UI:**

**Vault1:** [**http://54.145.131.93:8200/**](http://54.145.131.93:8200/)

**Vault2:** [**http://18.205.103.30:8200/**](http://18.205.103.30:8200/)

**Vault3:** [**http://34.238.120.59:8200/**](http://34.238.120.59:8200/)

**Vault4:** [**http://3.91.145.125:8200/**](http://3.91.145.125:8200/)

**Reference Link**

[**https://learn.hashicorp.com/tutorials/vault/raft-storage**](https://learn.hashicorp.com/tutorials/vault/raft-storage)

**USER CONFIGURATION USING CURL**

**We need to have admin policy in place:**

**adminPolicy**

# Manage auth backends broadly across Vault

path "auth/\*"

{

capabilities = ["create", "read", "update", "delete", "list", "sudo"]

}

# List, create, update, and delete auth backends

path "sys/auth/\*"

{

capabilities = ["create", "read", "update", "delete", "sudo"]

}

# To list policies - Step 3

path "sys/policy"

{

capabilities = ["read"]

}

# Create and manage ACL policies broadly across Vault

path "sys/policy/\*"

{

capabilities = ["create", "read", "update", "delete", "list", "sudo"]

}

# List, create, update, and delete key/value secrets

path "secret/\*"

{

capabilities = ["create", "read", "update", "delete", "list", "sudo"]

}

# Manage and manage secret backends broadly across Vault.

path "sys/mounts/\*"

{

capabilities = ["create", "read", "update", "delete", "list", "sudo"]

}

# Read health checks

path "sys/health"

{

capabilities = ["read", "sudo"]

}

# To perform Step 4

path "sys/capabilities"

{

capabilities = ["create", "update"]

}

# To perform Step 4

path "sys/capabilities-self"

{

capabilities = ["create", "update"]

}

**Create user using curl command**

**Create a payload.json with password:**

ubuntu@ip-172-31-87-119:~$ cat pavanlogin.json

{

"password": "pavan"

“policies” : “admin,default”

}

**Curl command:**

curl --header "X-Vault-Token:s.9ywq6OhmUyrGxl3e6m74dRbp" \

--request POST \

--data @pavanlogin.json \

http://3.94.98.140:8200/v1/auth/userpass/users/pavan

Reference Link : <https://www.vaultproject.io/api/auth/userpass>

**Login using userpass auth with curl command**

**Curl command to login as user:**

create a json file with the password for the user

**login.json**

{

"password": "pavan"

}

**Command :**

curl \

--request POST \

--data @pavanlogin.json \

<http://1c05353a-default-vault-c083-1304294467.ap-southeast-1.elb.amazonaws.com/v1/auth/userpass/login/pavan>

**output:**

{"request\_id":"8a89d883-ab30-56c7-559b-62f2e545f40c","lease\_id":"","renewable":false,"lease\_duration":0,"data":null,"wrap\_info":null,"warnings":null,"auth":{"client\_token":"**s.2KBgVYJA157XbhBxo5QaFAXN**","accessor":"ZlF33H0OrZTHrAs00aEwcliB","policies":["applepolicy","default"],"token\_policies":["default"],"identity\_policies":["applepolicy"],"metadata":{"username":"pavan"},"lease\_duration":2764800,"renewable":true,"entity\_id":"5de92794-4421-603a-b5db-16f1b791cda7","token\_type":"service","orphan":true}}

Note :

every time a user logs in using curl a new client\_token will be generated he can use this token as **X-Vault-Token:** s.2KBgVYJA157XbhBxo5QaFAXN **"**

**Generating aws ec2role credentials using new client-token:**

Pavan has access to ec2 role to generate credentials by using the new client token we can use the following command.

curl --header "X-Vault-Token: s.2KBgVYJA157XbhBxo5QaFAXN" <http://1c05353a-default-vault-c083-1304294467.ap-southeast-1.elb.amazonaws.com/v1/aws/creds/ec2role>

**output:**

{"request\_id":"7b5e8867-d946-65a5-ca03-bc34b33e8b2a","lease\_id":"aws/creds/ec2role/sKmw8mbYLkZxpCCGRdvYkify","renewable":true,"lease\_duration":1800,"data":{"access\_key":"AKIATSRJXG253T774U42","secret\_key":"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","security\_token":null},"wrap\_info":null,"warnings":null,"auth":null}

**List the roles in aws using the new token:**

curl \

--header "X-Vault-Token: s.2KBgVYJA157XbhBxo5QaFAXN" \

--request LIST \

http://1c05353a-default-vault-c083-1304294467.ap-southeast-1.elb.amazonaws.com/v1/aws/roles

**output:**

{"request\_id":"e3ee5b64-ae33-35dd-fa95-408e5e86984f","lease\_id":"","renewable":false,"lease\_duration":0,"data":{"keys":["ec2role","eksaccess","mys3role","s3role"]},"wrap\_info":null,"warnings":null,"auth":null}

Note :

The generated token will be vaild until the lease gets expired or until you revoke the token manually.