**PART-1**

**Creating a network layout & Deploying 3 node vault cluster**

**STEP-1: CREATING NETWORK LAYOUT AND DEPLOYING VAULT CLUSTER**

For this, we apply the terraform script which is in place that creates vpc, 3 subnets in 3 availability zones & deploys three vault servers, one vault transit server on 3 different public subnets in three different availability zones.

cd vault-newrk

terraform init

terraform plan

terraform apply –auto-approve

After this you will have your vault servers & vault transit deployed onto each AZ in given region.

**STEP-2: JOINING INDIVIDUAL VAULTS TO CLUSTER**

* Initially, **vault\_2** is initialized, unsealed, and has HA mode enabled. It is the only node in a cluster.
* The remaining nodes, **vault\_3** and **vault\_4**, have not joined its cluster.
* We will manually join the **vault\_3** and **vault\_4** to the cluster.

Go to vault-2 server and copy root token from ~root\_token/

Go to vault-3 server and perform operator join and then login into the vault server and list peers which shows the leader and its corresponding followers.

vault operator raft join http://174.129.52.216:8200

vault login

vault operator raft list-peers

Go to vault-4 server and perform operator join and then login into the vault server and list peers which shows the leader and its corresponding followers.

vault operator raft join http://174.129.52.216:8200

vault login

vault operator raft list-peers

Now your three node vault cluster is in place and you can play around it.

**PART-2**

**ENABLE USERPASS AUTH, CREATE ADMIN POLICY & CREATE USER WITH THE ADMIN POLICY**

**ENABLE USERPASS AUTH METHOD:**

The ***userpass*** auth method allows users to authenticate with Vault using a username and password combination.

Enable the ***userpass***using api as follows:

curl \

--header "X-Vault-Token: s.oYBa3IrZnMMwXWDBB1Uh491r " \

--request POST \

--data '{"type": "userpass"}' \

http:// 174.129.52.216:8200/v1/sys/auth/userpass

**CREATION OF ADMIN POLICY:**

An **admin** user must be able to:

* Read system health check
* Create and manage ACL policies broadly across Vault
* Enable and manage authentication methods broadly across Vault
* Manage the Key-Value secrets engine enabled at secret/ path
* Manage the aws secrets engine which we are going to enable at path -

**adminpolicy.hcl**

# 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"]

}

path "sys/mounts/\*"

{

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

}

# Read health checks

path "sys/health"

{

capabilities = ["read", "sudo"]

}

path "sys/capabilities"

{

capabilities = ["create", "update"]

}

path "sys/capabilities-self"

{

capabilities = ["create", "update"]

}

path “dynamic-aws-creds-vault-admin-path/\*”

{

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

}

To create policy using api you need to change your policy.hcl to payload.json as follows:

**adminPolicy.json:**

{

"path": {

"auth/\*": {

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

},

"sys/auth/\*": {

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

},

"sys/policy": {

"capabilities": ["read"]

},

"sys/policy/\*": {

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

},

"secret/\*": {

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

},

"sys/mounts": {

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

},

"sys/mounts/\*": {

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

},

"sys/health": {

"capabilities": ["read", "sudo"]

},

"dynamic-aws-creds-vault-admin-path/\*":{

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

}

}

}

policy\_string=$(cat vault/policies/adminPolicy.json | jq -c . | sed 's/"/\\\"/g')

curl -X PUT -H "X-Vault-Token: s.IZeqoi10Pez65z1hEqHvbiWb " -d "{\"policy\":\"${policy\_string}\"}" <http://52.90.88.15:8200/v1/sys/policy/adminpolicy>

**Reference link:**

<https://github.com/hashicorp/vault/issues/582>

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

{

"policy": "# Manage auth backends broadly across Vault\npath \"auth/\*\"\n{\n capabilities = [\"create\", \"read\", \"update\", \"delete\", \"list\", \"sudo\"]\n}\n\n# List, create, update, and delete auth backends\npath \"sys/auth/\*\"\n{\n capabilities = [\"create\", \"read\", \"update\", \"delete\", \"sudo\"]\n}\n\n# List existing policies\npath \"sys/policy\"\n{\n capabilities = [\"read\"]\n}\n\n# Create and manage ACL policies broadly across Vault\npath \"sys/policy/\*\"\n{\n capabilities = [\"create\", \"read\", \"update\", \"delete\", \"list\", \"sudo\"]\n}\n\n# List, create, update, and delete key/value secrets\npath \"secret/\*\"\n{\n capabilities = [\"create\", \"read\", \"update\", \"delete\", \"list\", \"sudo\"]\n}\npath \"dynamic-aws-creds-vault-admin-path/\*\"\n{\n capabilities = [\"create\", \"read\", \"update\", \"delete\", \"list\", \"sudo\"]\n}\n# Manage and manage secret backends broadly across Vault.\npath \"sys/mounts\"\n{\n capabilities = [\"create\", \"read\", \"update\", \"delete\", \"list\", \"sudo\"]\n}\npath \"sys/mounts/\*\"\n{\n capabilities = [\"create\", \"read\", \"update\", \"delete\", \"list\", \"sudo\"]\n}\n\n# Read health checks\npath \"sys/health\"\n{\n capabilities = [\"read\", \"sudo\"]\n}"

}

Use the following Command to Create Admin Policy:

curl --request PUT --header "X-Vault-Token: s.oYBa3IrZnMMwXWDBB1Uh491r " --data @admin-payload.json http://174.129.52.216:8200/v1/sys/policy/admin

**CREATION OF USER WITH ADMIN POLICY:**

Now, we need to assign the admin policy which we created earlier to an user as follows:

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

{

"password": "pavan",

"policies": "admin"

}

curl --header "X-Vault-Token:s.oYBa3IrZnMMwXWDBB1Uh491r " \

--request POST \

--data @pavanuser.json \

<http://174.129.52.216:8200/v1/auth/userpass/users/pavan>

**Login with user to get client token :**

curl --header "X-Vault-Token:s.oYBa3IrZnMMwXWDBB1Uh491r " \

--data @pavanlogin.json \

<http://174.129.52.216:8200/v1/auth/userpass/login/pavan>

**output:**

{"request\_id":"319967f9-4045-f538-93b2-c2344a47cec8","lease\_id":"","renewable":false,"lease\_duration":0,"data":null,"wrap\_info":null,"warnings":null,

"auth":{"client\_token":"**s.oXvOfxu0S6KZ0yWpkCHMvemB**","accessor":"pCFte0TZXPdLR9ti0KPwIme3",

"policies":["admin","default"],"token\_policies":["admin","default"],

"metadata":{"username":"pavan"},"lease\_duration":2764800,"renewable":true,"entity\_id":"244903b8-af6f-04ec-b34e-50e594200c4b","token\_type":"service","orphan":true}}

**PART-3 BACKEND CREATION FOR AWS**

**INJECT SECRETS INTO TERRAFORM USING THE VAULT PROVIDER**

* First, as a Vault Admin, you will configure AWS Secrets Engine in Vault.
* Then, as a Terraform Operator, you will connect to the Vault instance to retrieve dynamic, short-lived AWS credentials generated by the AWS Secrets Engine to provision an Ubuntu EC2 instance and S3 bucket creation.

**EXPORT VAULT ADDRESS AND TOKEN:**

we're setting the required [Vault Provider arguments](https://www.terraform.io/docs/providers/vault/index.html#provider-arguments) as environment variables: VAULT\_ADDR & VAULT\_TOKEN.

export VAULT\_TOKEN=s.oXvOfxu0S6KZ0yWpkCHMvemB

export VAULT\_ADDR=http://174.129.52.216:8200/

Here we will export the client token generated for pavan user instead of using root-token directly.

**VAULT ADMIN:**

Perform “terraform apply” inside **vault-adminspace**

This will enable secret engine in path dynamic-aws-creds-vault-admin-path

Then 3 roles will be created for s3 bucket, ec2 access and eks access.

**TERRAFORM OPERATOR:**

Now that we have successfully configured Vault's AWS Secrets Engine, you can retrieve dynamic short lived AWS credentials to provision an EC2 instance and S3 bucket.

Perform “terraform apply” inside **operator-workspace**

In directory **operator workspace ** the terraform script in this path will read dynamic credentials from dynamic-aws-creds-vault-admin-path for ec2 role using which an ec2 instance will be launched.

Perform “terraform apply” inside **operator-workspace/s3creation** workspace

Navigate to the [***IAM Users page***](https://console.aws.amazon.com/iam/home?region=us-east-1#/users) in AWS Console. Search for the username prefix vault-token-terraform-dynamic-aws-creds-vault-admin. You should see a IAM user.

This IAM user was generated by Vault with the appropriate IAM policy configured by the Vault Admin workspace. Because the default\_lease\_ttl\_seconds is set to 120 seconds, Vault will revoke those IAM credentials and they will be removed from the AWS IAM console after 120 seconds.

In directory **operator workspace  s3creation ** the terraform script in this path will read dynamic credentials from dynamic-aws-creds-vault-admin-path for s3 role using which an s3 bucket will be created.