**Vault Integration**

Here are the detailed steps for each of these steps:

**Create an AWS EC2 instance with Ubuntu**

To create an AWS EC2 instance with Ubuntu, you can use the AWS Management Console or the AWS CLI. Here are the steps involved in creating an EC2 instance using the AWS Management Console:

* Go to the AWS Management Console and navigate to the EC2 service.
* Click on the Launch Instance button.
* Select the Ubuntu Server xx.xx LTS AMI.22
* Select the instance type that you want to use.
* Configure the instance settings.
* Click on the Launch button.

**Install Vault on the EC2 instance**

To install Vault on the EC2 instance, you can use the following steps:

**Install gpg**

sudo apt update && sudo apt install gpg

**Download the signing key to a new keyring**

wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

**Verify the key's fingerprint**

gpg --no-default-keyring --keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg --fingerprint

**Add the HashiCorp repo**

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb\_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt update

**Finally, Install Vault**

sudo apt install vault

**Start Vault.**

To start Vault, you can use the following command:

vault server -dev -dev-listen-address="0.0.0.0:8200"

**Configure Terraform to read the secret from Vault.**

Detailed steps to enable and configure AppRole authentication in HashiCorp Vault:

1. **Enable AppRole Authentication**:

To enable the AppRole authentication method in Vault, you need to use the Vault CLI or the Vault HTTP API.

**Using Vault CLI**:

Run the following command to enable the AppRole authentication method:

**vault auth enable approle**

This command tells Vault to enable the AppRole authentication method.

1. **Create an AppRole**:

We need to create policy first,

vault policy write terraform - <<EOF

path "\*" {

capabilities = ["list", "read"]

}

path "secrets/data/\*" {

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

}

path "kv/ data /\*" {

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

}

path "kv/db\_cred/\*" {

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

}

path "secret/data/\*" {

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

}

path "auth/token/create" {

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

}

EOF

Now you'll need to create an AppRole with appropriate policies and configure its authentication settings. Here are the steps to create an AppRole:

**a. Create the AppRole**:

vault write auth/approle/role/terraform \

secret\_id\_ttl=10m \

token\_num\_uses=10 \

token\_ttl=20m \

token\_max\_ttl=30m \

secret\_id\_num\_uses=40 \

token\_policies=terraform

1. **Generate Role ID and Secret ID**:

After creating the AppRole, you need to generate a Role ID and Secret ID pair. The Role ID is a static identifier, while the Secret ID is a dynamic credential.

**a. Generate Role ID**:

You can retrieve the Role ID using the Vault CLI:

vault read auth/approle/role/terraform/role-id

Save the Role ID for use in your Terraform configuration.

**b. Generate Secret ID**:

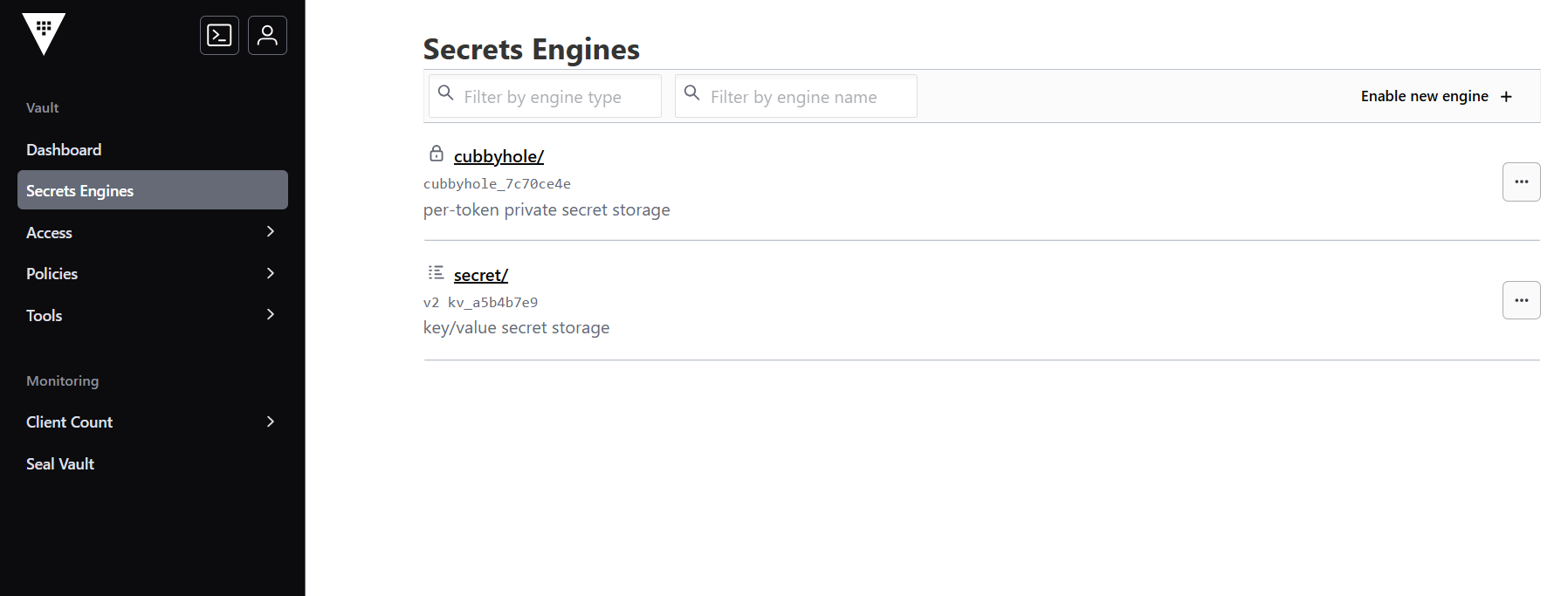
To generate a Secret ID, you can use the following command:

vault write -f auth/approle/role/terraform/secret-id

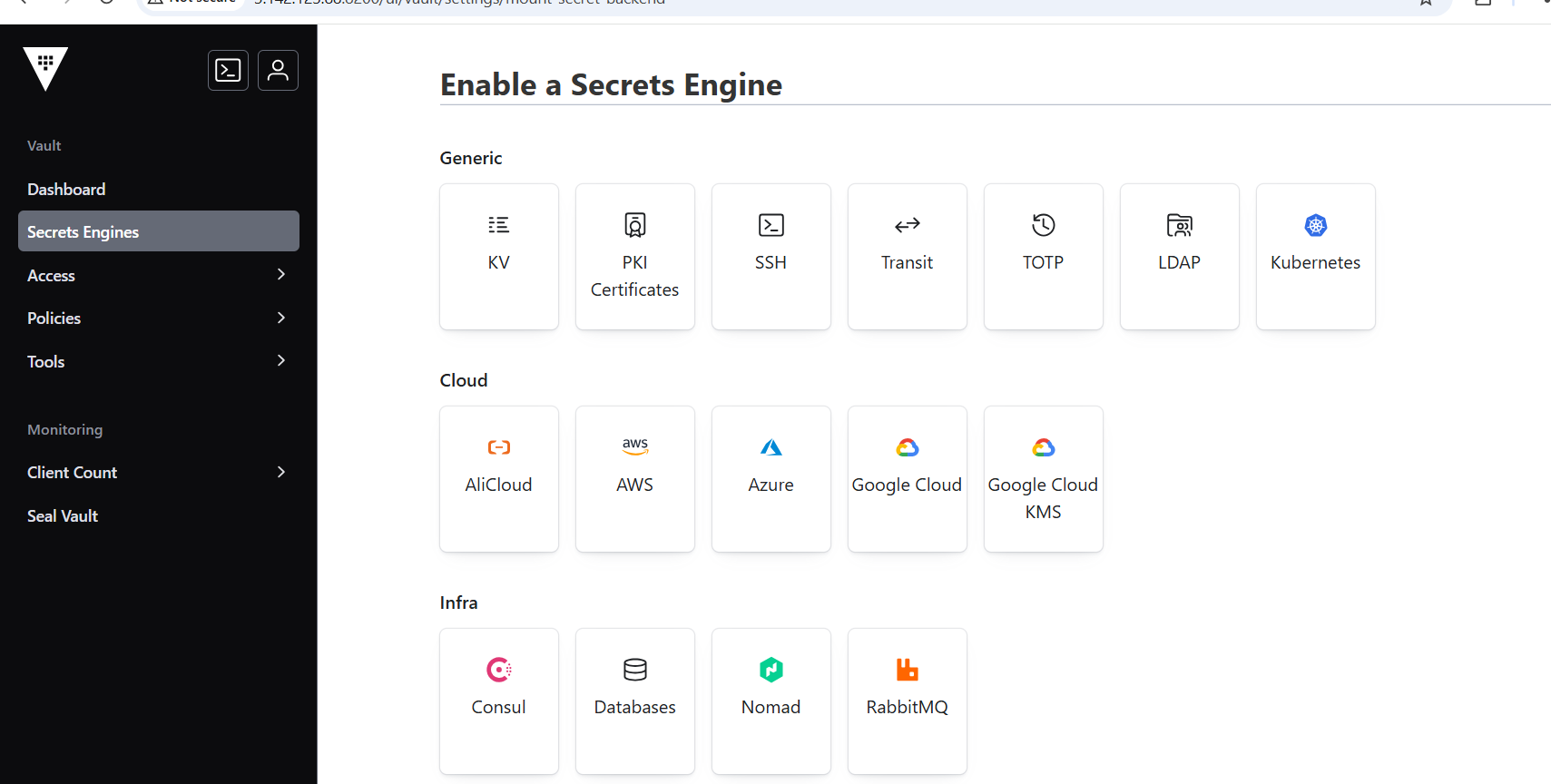
This command generates a Secret ID and provides it in the response. Save the Secret ID securely, as it will be used for Terraform authentication.

Add a secret on terraform Vault:

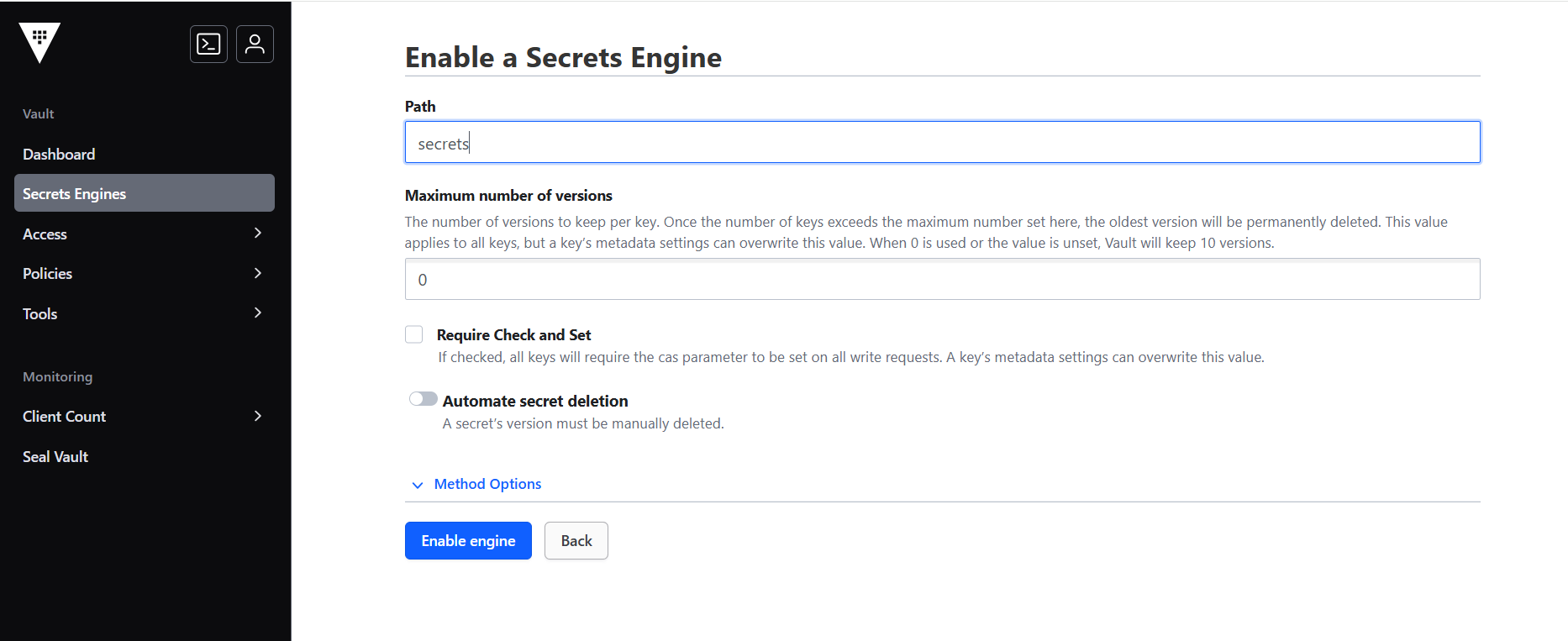
Click on Secrets Engine -> click on enable new engine



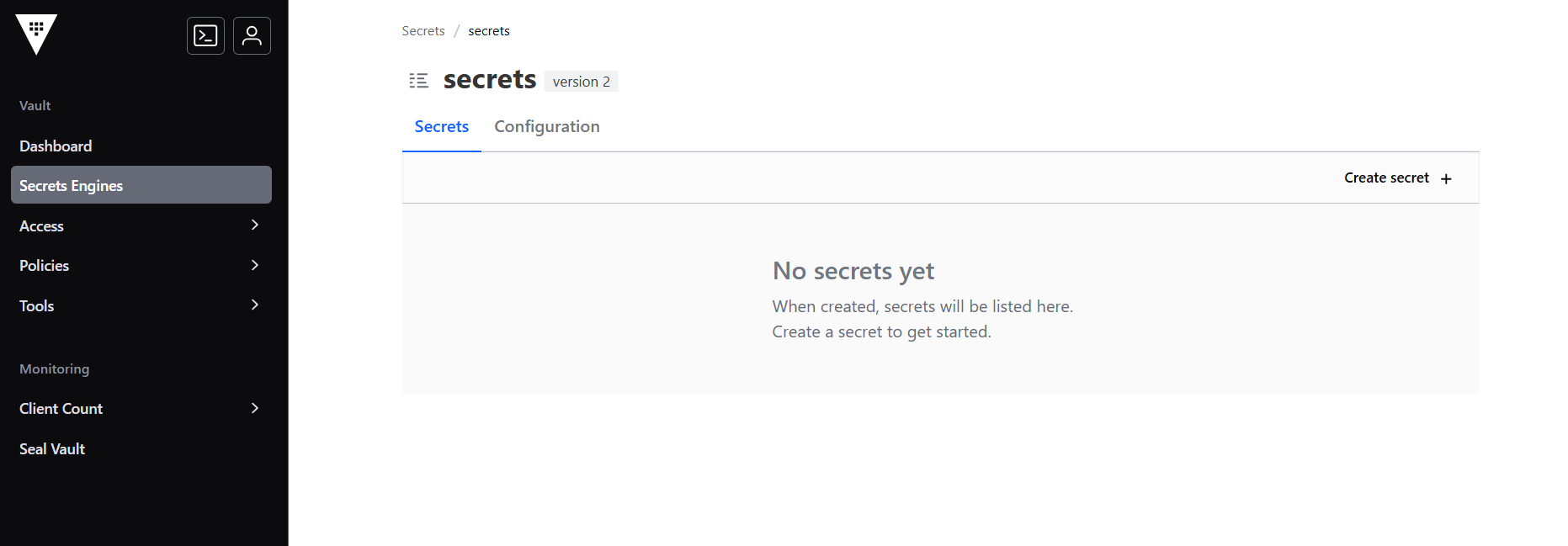
Click on KV



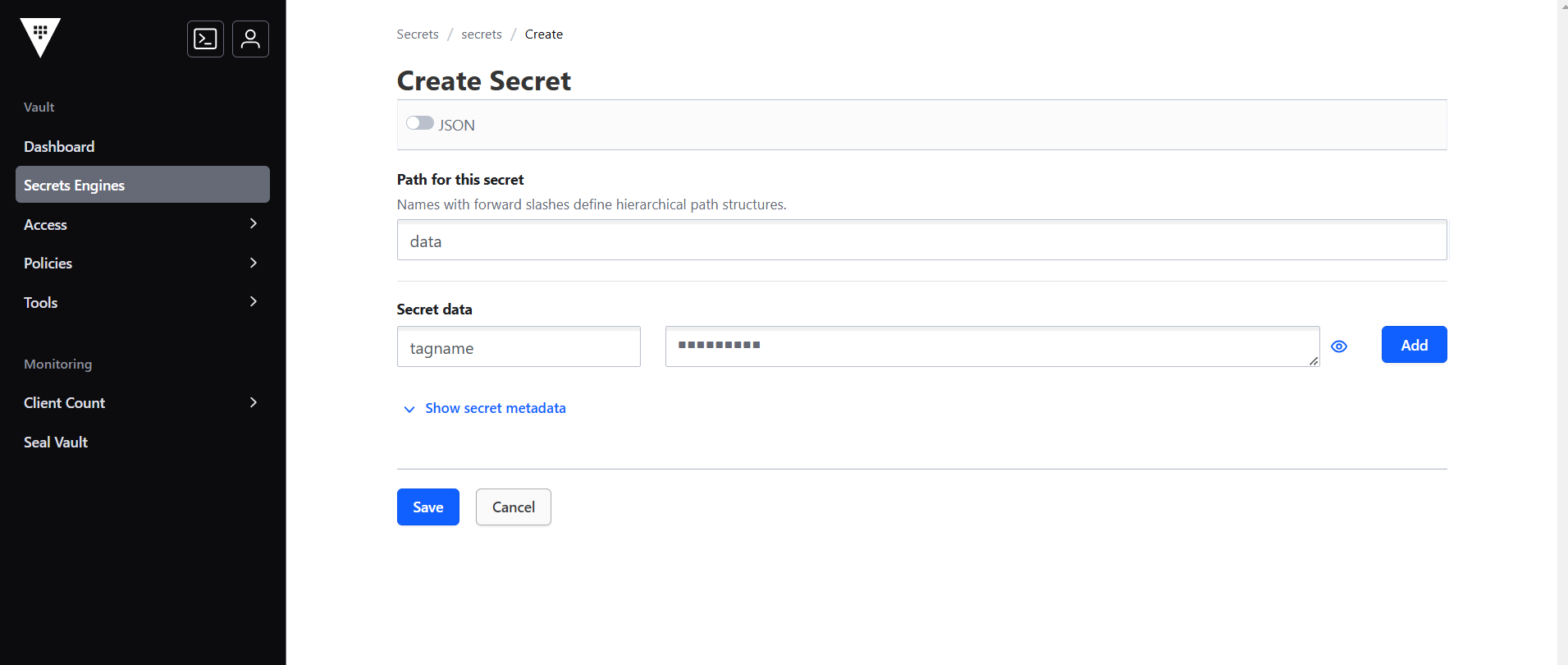
Give path as secrets – give same



Click on enable engine and click on Create secret



Give the same value



Create Role in Vault CLI

Demo:

provider "aws" {

region = "us-east-1"

}

provider "vault" {

address = "<>:8200"

skip\_child\_token = true

auth\_login {

path = "auth/approle/login"

parameters = {

role\_id = "<>"

secret\_id = "<>"

}

}

}

data "vault\_kv\_secret\_v2" "example" {

mount = "secret" // change it according to your mount

name = "test-secret" // change it according to your secret

}

resource "aws\_instance" "my\_instance" {

ami = "ami-053b0d53c279acc90"

instance\_type = "t2.micro"

tags = {

Name = "test"

Secret = data.vault\_kv\_secret\_v2.example.data["foo"]

}

}

Demo 2:

provider "vault" {

address = "http://3.85.114.238:8200"

skip\_child\_token = true

auth\_login {

path = "auth/approle/login"

parameters = {

role\_id = "18e473ff-d6a8-4f45-0ba9-ab35e68ac01a"

secret\_id = "ae852c53-0ed5-a619-8457-37ae3e8df266"

}

}

}

data "vault\_kv\_secret\_v2" "vaultdata" {

mount = "kv" // change it according to your mount

name = "db\_cred" // change it according to your secret

}

resource "aws\_db\_instance" "default" {

db\_name = "mydb"

engine = "mysql"

engine\_version = "5.7"

instance\_class = "db.t2.micro"

username = data.vault\_kv\_secret\_v2.vaultdata.data["username"]

password = data.vault\_kv\_secret\_v2.vaultdata.data["password"]

parameter\_group\_name = "default.mysql5.7"

skip\_final\_snapshot = true

}