Day 3 – Terraform

- **hashicorp vault or terraform wall**. Both are same. So you install terraform from the hashicorp page. Only so hashicorp is an organization. So even the terraform belongs to hashicorp. You all to also belong to hashicorp.

    1. **remote back end**
    2. **environmental variable**
    3. **aws system manager**
    4. **sensitive attribute**
    5. **credentials**
    6. **encryption**
    7. **audit locks**

Remote backend:

- **we use a wall to store the sensitive data that we that gets involved in your terraform file**. So, there are different ways where you could manage the sensitive data.
- 1st thing, how you secure your state by using the remote back end. So, using the backend block, you save your state file remotely, so what will happen. It will protect your state file so it will secure your state file. when the State file is maintained, secure in the remote.

Environmental variable:

- So, you're passing the access key and secret access key before initializing the terraform.
- So even your access key and secret access key can be maintained in your terraform vault, and from the vault you will be using it inside your terraform file.
- Whenever that terraform is getting initialized, it will 1st take the access key and secret access key from the vault, and it will initialize the terraform file so you can. You can keep your keys there in your you can maintain your keys in your terraform world as well.

Aws system manager /secret manager :

- So you can make use of these services to store the sensitive data for your terraform file .So these are some of the methods that you can store the sensitive data.

Sensitive attribute:

- So when you mark an attribute as a sensitive attribute, what will happen this this variable or the value will not be printed on the terminal.
- Okay, it will be. It will be hidden, or you cannot see. The values will not be exposed in your terminal file. That is in your terminal screen. When you give output in your output block you, you'll you will give. You wanted to get the output of certain values like, for like your instance, Id, your s3 bucket id, and all those things, you put it inside your output block.
- while in the output block certain things. If you don't want to get exposed in the console output that you can mark it as a sensitive, you can mark the variable with a sensitive attribute.
- So what will happen? Those variables will be those variables value will not be exposed on the console.
- So now we will make use of this terraform vault to store the sensitive data. So using terraform vault, what all you can do this.

Credentials:

- so you can store the access key and secret access key. You can store your credentials. It can be any credentials, your aws credential, your azure credentials, any credentials you can save so mainly to store your credentials, that is, to store your sensitive data and using what you can also dynamically generate the credentials. So you can use dynamic secrets.
- So what will happen, it will generate on demand secrets. So for your databases or for any of your cloud services you want to generate any keys you can make use of this vault, and your sensitive data will also be encrypted.

Encryption:

- So data encryption is possible using your vault. Okay, so data will be encrypted. Your sensitive data will be encrypted during your transit. And also, during the time of rest. Okay, when the data is that when you're putting your data inside your vault. The data is at risk. Okay, even during that time your data is encrypted.
- When you're moving your data from the vault to your terraform file , then your data is in transit. Even during that time your data will encrypt.

Audit locks :

- so you can get the complete audit logs of all your access and the secret retrieval. So your vault will have the complete log for accessing.
- If all the logs will be maintained by your vault.

Why we use vault:

- Store sensitive data
- API keys
- Certificates
- Passwords

Where use that vault in terraform file:

- how do you retrieve it? So you have a keyword called data.
- So you have a keyword called data. Using this data keyword, you can get, you can retrieve the secrets from the vault into your terraform file. So we will be in the terraform file.
- We used resource block. We used output block and we use Provider Block. So in your resource block what you're doing. You're creating a resource or a service in your Aws account.
- Then you will be creating a block called data. So, this data block will help you to retrieve the secret from your retrieve the secret from the vault . So here we will be mentioning this data block inside our terraform files.
- So, in your terraform will use a data block to retrieve the secret from the vault.

how to install vault :

https://developer.hashicorp.com/vault/tutorials/getting-started/getting-started-install

doc link for install vault

Install Vault

Manual macOS Windows **Linux**

HashiCorp officially maintains and signs packages for the following Linux distributions.

Ubuntu/Debian CentOS/RHEL Fedora Amazon Linux

Please follow the instructions in the Official Packaging Guide to install the HashiCorp GPG key, verify the key's fingerprint, and install Vault.

Update the package manager and install GPG and wget.

```
$ sudo apt update && sudo apt install gpg wget
```

Download the keyring

```
$ wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o r/
```

Copy and paste the cmd one by one

```
ubuntu@ip-172-31-6-100:~$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases
.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
deb [arch=amd64 signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com jammy main
ubuntu@ip-172-31-6-100:~$ sudo apt update && sudo apt install vault
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 https://apt.releases.hashicorp.com jammy InRelease
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
6 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
vault is already the newest version (1.17.2-1).
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
ubuntu@ip-172-31-6-100:~$
```

Terraform and vault installed in same machine.

you can start a vault server as a production server or a development server. Now, now, since we are using our Ec 2, we can go with the development server. So now we will be starting our vault as a development server.

Walters, a UI based application like your Jenkins. So after installing vault, we can access vault using the Ui and vaults on the port number 8200. Give the below cmd

```
ubuntu@ip-172-31-6-100:~$ vault server -dev -dev-listen-address="0.0.0.0:8200"
==> Vault server configuration:

Administrative Namespace:
             Api Address: http://0.0.0.0:8200
                     Cgo: disabled
         Cluster Address: https://0.0.0.0:8201
```

```
WARNING! dev mode is enabled! In this mode, Vault runs entirely in-memory
and starts unsealed with a single unseal key. The root token is already
authenticated to the CLI, so you can immediately begin using Vault.

You may need to set the following environment variables:

    $ export VAULT_ADDR='http://0.0.0.0:8200'

The unseal key and root token are displayed below in case you want to
seal/unseal the Vault or re-authenticate.

Unseal Key: dt06yI/aCxYjSzF+68kzp8hOX8L+27RoH2cuJH5D8II=
Root Token: hvs.H9sGJ8hysAizqvOFCCsZu00D

Development mode should NOT be used in production installations!
```

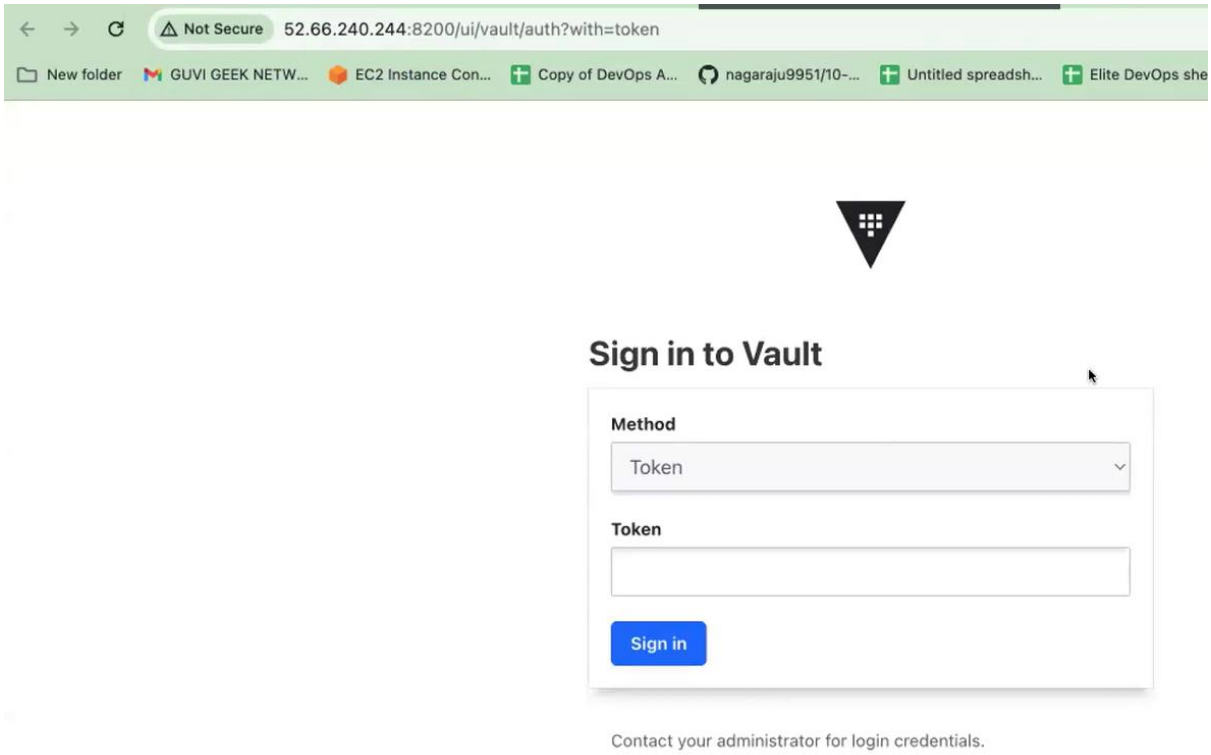
Duplicate the window give the marked variable

So now you have exported

```
Last login: Mon Jul 15 11:21:30 2024 from 13.233.177.3
ubuntu@ip-172-31-6-100:~$ export VAULT_ADDR='http://0.0.0.0:8200'
ubuntu@ip-172-31-6-100:~$
```

I have to access the vault now :

So to access your vault, just go to your Ec. 2. You're in this machine [demo-tf] only I have installed vault. Let me copy the IP address. So I have already opened my security group in this machine. So you have to open up the port number 8200. So in this port number only your vault is running.



Running cmd vault server -dev that time we got this token password.

```
Unseal Key: dt06yI/aCxYjSzF+68kzp8hOX8L+27RoH2cuJH5D8II=
Root Token: hvs.H9sGJ8hysAizqvOFCCsZu00D
```

Sign in to Vault

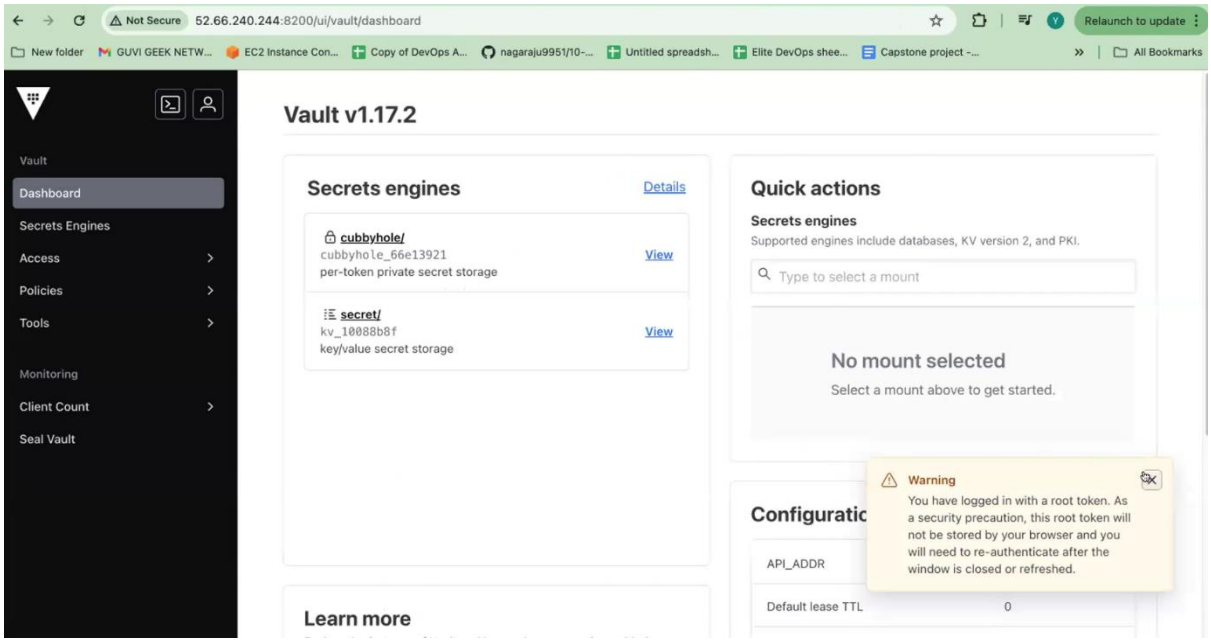
Method

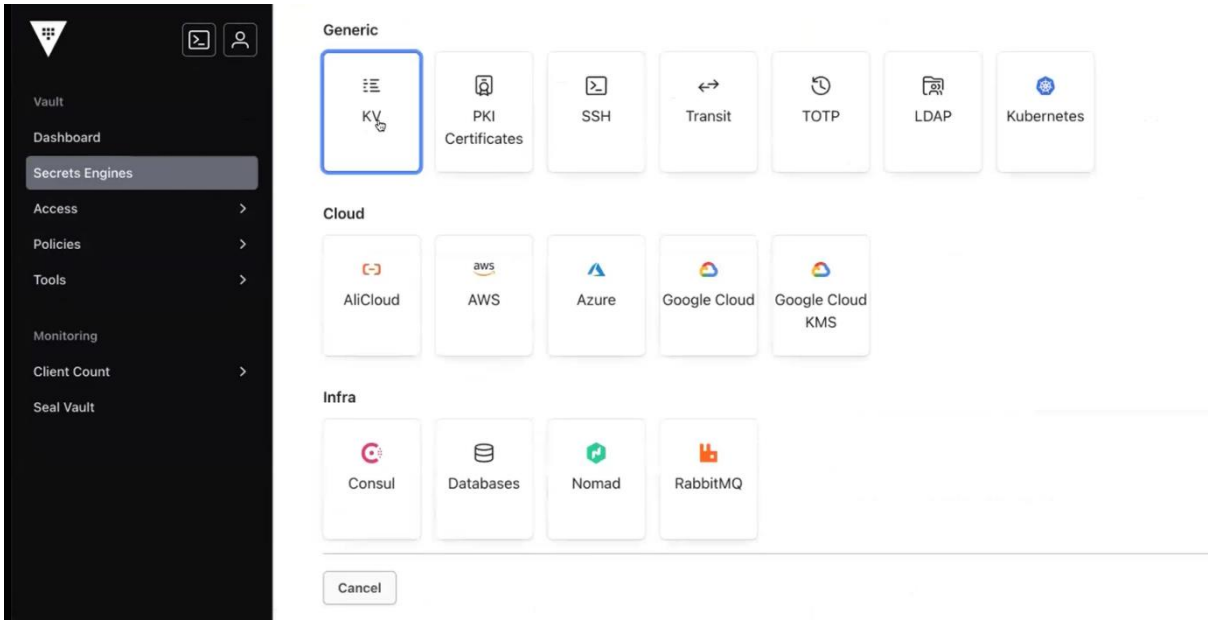
Token

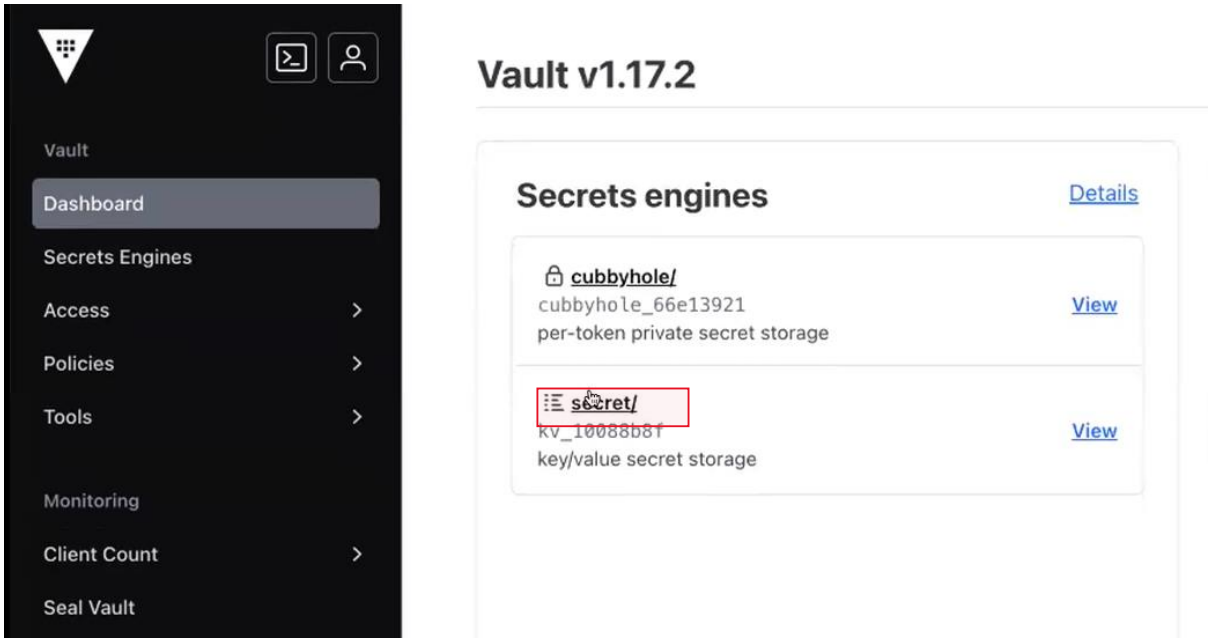
Token

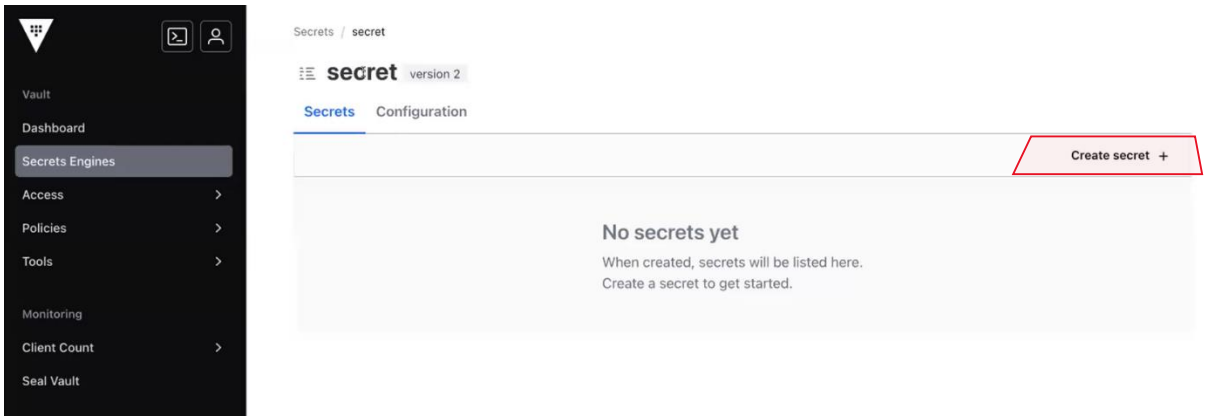
Sign in

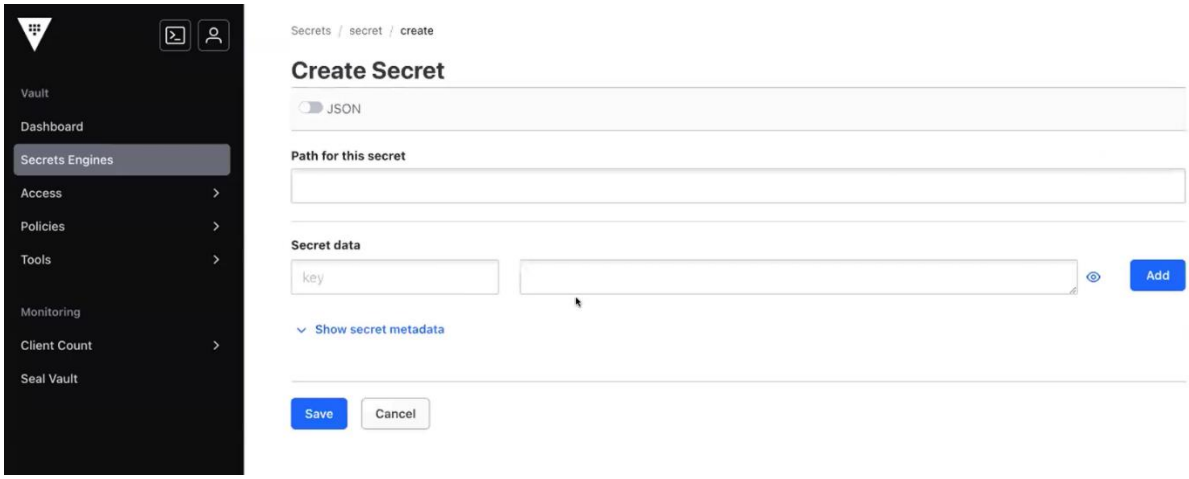
Contact your administrator for login credentials.


Not Secure 52.66.240.244:8200/ui/vault/dashboard
Relaunch to update
Vault v1.17.2
Vault
Dashboard
Secrets Engines
Access
Policies
Tools
Monitoring
Client Count
Seal Vault
Secrets engines
Details
cubbyhole/
cubbyhole_66e13921
per-token private secret storage
View
secret/
kv_10088b8f
key/value secret storage
View
Quick actions
Secrets engines
Supported engines include databases, KV version 2, and PKI.
Type to select a mount
No mount selected
Select a mount above to get started.
Warning
You have logged in with a root token. As a security precaution, this root token will not be stored by your browser and you will need to re-authenticate after the window is closed or refreshed.
API_ADDR
Default lease TTL
0
Learn more

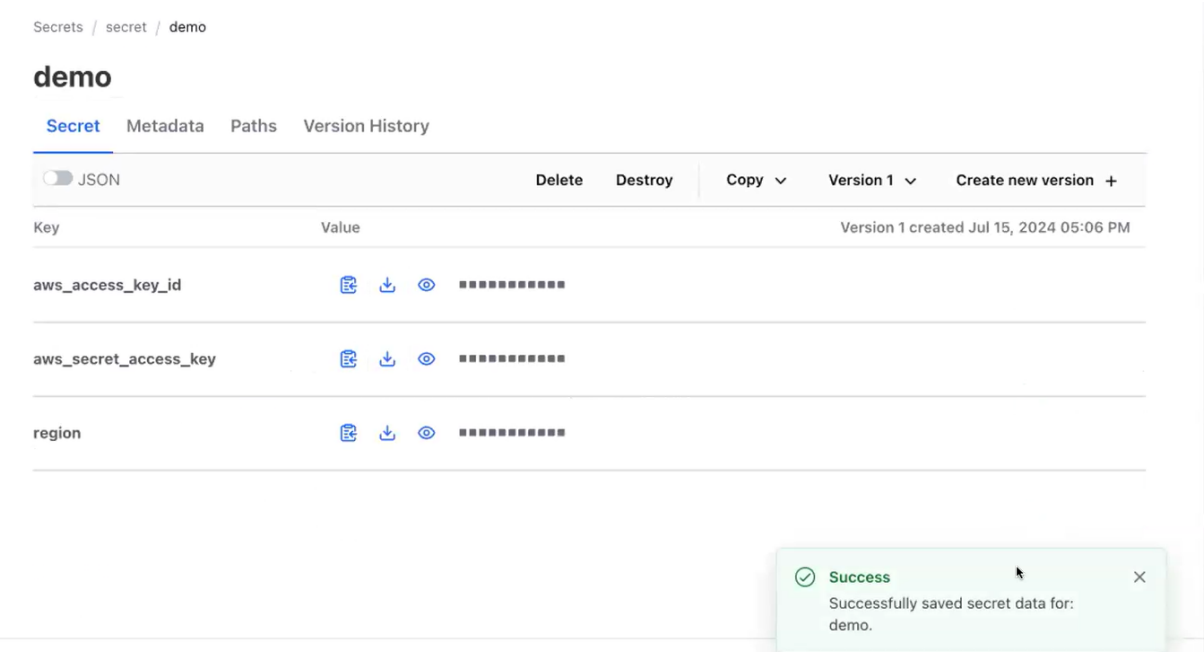

Vault
Dashboard
Secrets Engines
Access
Policies
Tools
Monitoring
Client Count
Seal Vault
Generic
KV
PKI Certificates
SSH
Transit
TOTP
LDAP
Kubernetes
Cloud
AliCloud
AWS
Azure
Google Cloud
Google Cloud KMS
Infra
Consul
Databases
Nomad
RabbitMQ
Cancel







Add any secrets data here.



Create a directory ,

```
ubuntu@ip-172-31-6-100:~$ mkdir testvault
ubuntu@ip-172-31-6-100:~$ cd testvault/
ubuntu@ip-172-31-6-100:~/testvault$ ls
ubuntu@ip-172-31-6-100:~/testvault$ vi main.tf
```

Terraform block:

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}
```

Provider block:

```
data "vault_generic_secret" "aws_creds" {
  path = "secret/demo"

}
```

Region:

```
provider "aws" {
  region = data.vault_generic_secret.aws_creds.data["region"]
   access_key = data.vault_generic_secret.aws_creds.data["aws_access_key_id"]
   secret_key = data.vault_generic_secret.aws_creds.data["aws_secret_access_key"]
}
```

Resource block: [create any resources using the resource block]

create a simple Ec 2

```
resource "aws_instance" "demo" {
        ami = "ami-0c2af51e265bd5e0e"
        instance_type = "t2.micro"
}
```

Terraform commands:

1. **Terraform init**
2. **Terraform plan [ask vault access. So give vault URL in that page]**
3. **Terraform apply**

I will create another folder. Now I'll be creating another secret.

Let me go to my vault.

I will create another secret.