HashiCorp Vault with Terraform

🔧1. Launch an EC2 Instance with Ubuntu

Via AWS Management Console:

- Go to the [EC2 Dashboard].
- Click Launch Instance.
- Choose Ubuntu Server xx.xx LTS (e.g., 22.04 LTS).
- Select an instance type (e.g., t2.micro for testing).
- Configure network and storage settings as needed.
- Add security group rules (allow at least port 8200 for Vault, 22 for SSH).
- Launch the instance using an existing or new key pair.

2. Install Vault on Ubuntu EC2

SSH into your instance, then run:

a. Install GPG:

```
sudo apt update && sudo apt install -y gpg wget
```

b. Add the HashiCorp GPG key:

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

Add HashiCorp repository and install Vault:

```
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

sudo apt install vault
```

3. Start Vault Server in Dev Mode

For demo/testing (not for production):

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

-dev mode runs Vault in memory, enables the root token, and allows unauthenticated access. Great for learning/testing.

In a separate terminal, set the environment variable:

```
export VAULT_ADDR='http://<EC2-PUBLIC-IP>:8200'

export VAULT_TOKEN='root'
```

4. Enable AppRole Authentication and Policies

a. Enable AppRole:

```
vault auth enable approle
```

b. Create a Policy named terraform:

```
vault policy write terraform-app-policy - <<EOF

path "secret/data/*" {

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

}



path "auth/token/create" {

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

}

EOF
```

Create AppRole:

```
vault write auth/approle/role/terraform-role \

   secret_id_ttl=60m \

   token_ttl=20m \

   token_max_ttl=60m \

   token_policies="terraform-app-policy"
```

Get AppRole Credentials

Get Role ID:

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

Generate Secret ID:

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

Save these safely—they will be used in Terraform.

Store Secrets in Vault

vault kv put secret/aws-creds access_key="AKIAEXAMPLEKEY" secret_key="s3cr3tK3y987"

Terraform Configuration:

Create main.tf:

```
provider "vault" {

  address = "http://<public-ip>:8200"

}



data "vault_approle_auth_backend_login" "login" {

  role_id   = "e.g. 8a5c3ae1-...."

  secret_id = "e.g. e3b1c4a8-...."

}



provider "vault" {

  address = "http://<public-ip>:8200"

  token   = data.vault_approle_auth_backend_login.login.client_token

}



data "vault_generic_secret" "aws_secrets" {

  path = "secret/data/aws-creds"

}



output "aws_access_key" {
  value = data.vault_generic_secret.aws_secrets.data["access_key"]

}


output "aws_secret_key" {

  value = data.vault_generic_secret.aws_secrets.data["secret_key"]

}
```

⚠️ ***Replace <public-ip> with your EC2's IP, and use the actual Role ID and Secret ID you fetched.***

Run Terraform:

```
terraform init

terraform apply
```

Confirm that Vault secrets are pulled correctly into Terraform.

Final Output:

```
aws_access_key = "AKIAEXAMPLEKEY"

aws_secret_key = "s3cr3tK3y987"
```

Tips for Production:

Best Practice	***Reason***
Avoid dev mode	***Use proper storage, TLS, and authentication***
Use Vault Agent or AWS IAM login	***Better security than Role ID + Secret ID***
Store Role ID/Secret ID in environment variables	***Avoid hardcoding***
Use dynamic secrets (Vault AWS engine)	***Auto-expiry & rotation of credentials***
Use private networking or VPN	***Never expose Vault on public IP without security***

Optional: Dynamic Secrets Example (AWS)

Instead of static credentials, use Vault's dynamic secrets engine for AWS

```
data "vault_aws_access_credentials" "creds" {

  backend = "aws"

  role    = "my-role"

}



provider "aws" {

  access_key = data.vault_aws_access_credentials.creds.access_key

  secret_key = data.vault_aws_access_credentials.creds.secret_key

  region     = "us-west-2"

}
```

Summary

- ***Vault*** *helps securely manage secrets.*
- ***Terraform*** *can fetch secrets from Vault using the Vault provider.*
- *Best practice is to avoid hardcoding secrets and use dynamic secrets when possible.*