**Vault Integration with Terraform**

**# Hashicorp Vault wit Terraform**

Steps:

1. Create an EC2

2. Take Public Ip and login into server (ssh -i <path/to/private/key> ubuntu@public\_ip)

3. Adding Hashocorp vault package into server, by default it not there

```

sudo apt update && sudo apt install gpg

```

4. 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

```

5. Verify the key's fingerprint

```

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

```

6. 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

```

7. Update repo

```

sudo apt update

```

8. Install Vault

```

sudo apt install vault

```

Check: type "vault" in server, it works

**# Start Vault**

1. To start Vault, you can use the following command: (u can use dev or prod )

```

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

```

Take a new tab for better practice. To see errors messages effectively.

ssh -i <path/to/private/key> ubuntu@public\_ip

2. copy and export below command to another terminal

```

**export VAULT\_ADDR='http://0.0.0.0:8200'**

```

Here you have root token copy for login to vault.

3. Now you can access your vault site and also add inbound port 8200 (public\_ip:8200)

```

http://18.188.173.96:8200

```

Login into Hashicorp Vault using Token

4. Enable engine by giving path: kv or secret.

5. Now create a secret, Sceret\_path: test-secret, Secret data: username and Value: Prasad (this is confidential) then save.

6. Now create a policy (like in AWS IAM), run below command.

```

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 "secret/data/\*" {

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

}

path "auth/token/create" {

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

}

EOF

```

7. Now create a role: Run below command to create a role:

```

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

```

8. Generate Role ID and Secret ID: Run below command to get role and secret ids.

**Role ID**

```

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

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

```

**Sceret ID:**

```

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

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

```

**# Write Terraform code to integrate with Vault**

**main.tf:**

```

provider "aws"{

    region = "us-east-1"

}

provider "vault" {

  address = "http://ec2\_public\_ip:8200"

  skip\_child\_token = true

  auth\_login {

    path = "auth/approle/login"

   parameters = {

      role\_id   = "PASTE-ROLE-ID"

      secret\_id = "PASTE-SECRET-ID"

  }

}

#data Source to read the existing data

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

  mount = "kv"

  name  = "test-secret"

}

#Now craete a resource

resource "aws\_instance" "main"{

    ami = "PASTE-AMI-ID"

    instance\_type = "t3.micro"

    vpc\_secuirty\_group\_ids = ["PASTE-SG-ID"]

    tags = {

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

    }

}

```

1. Run terraform commands

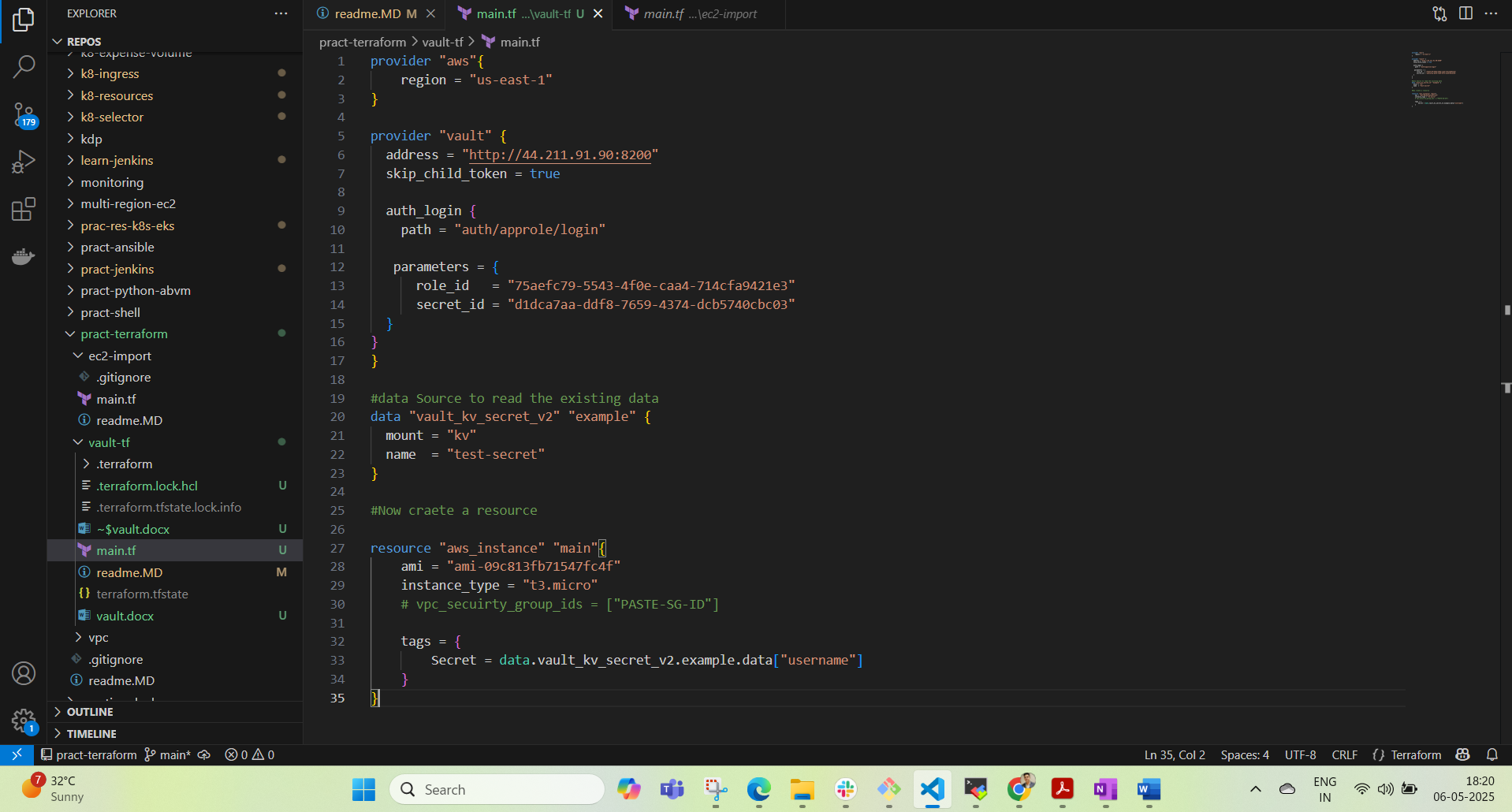
```

terraform init

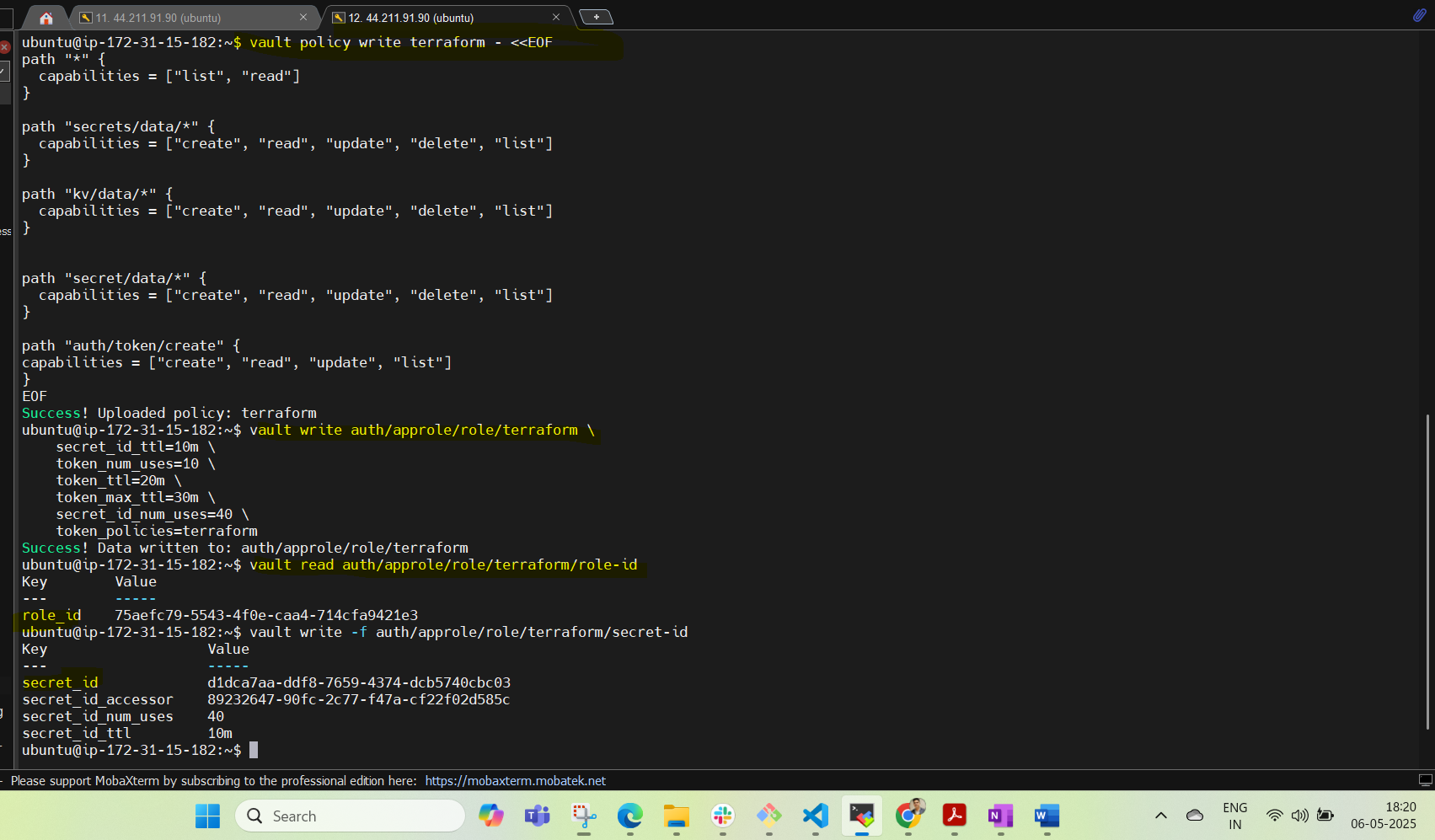
terraform paln

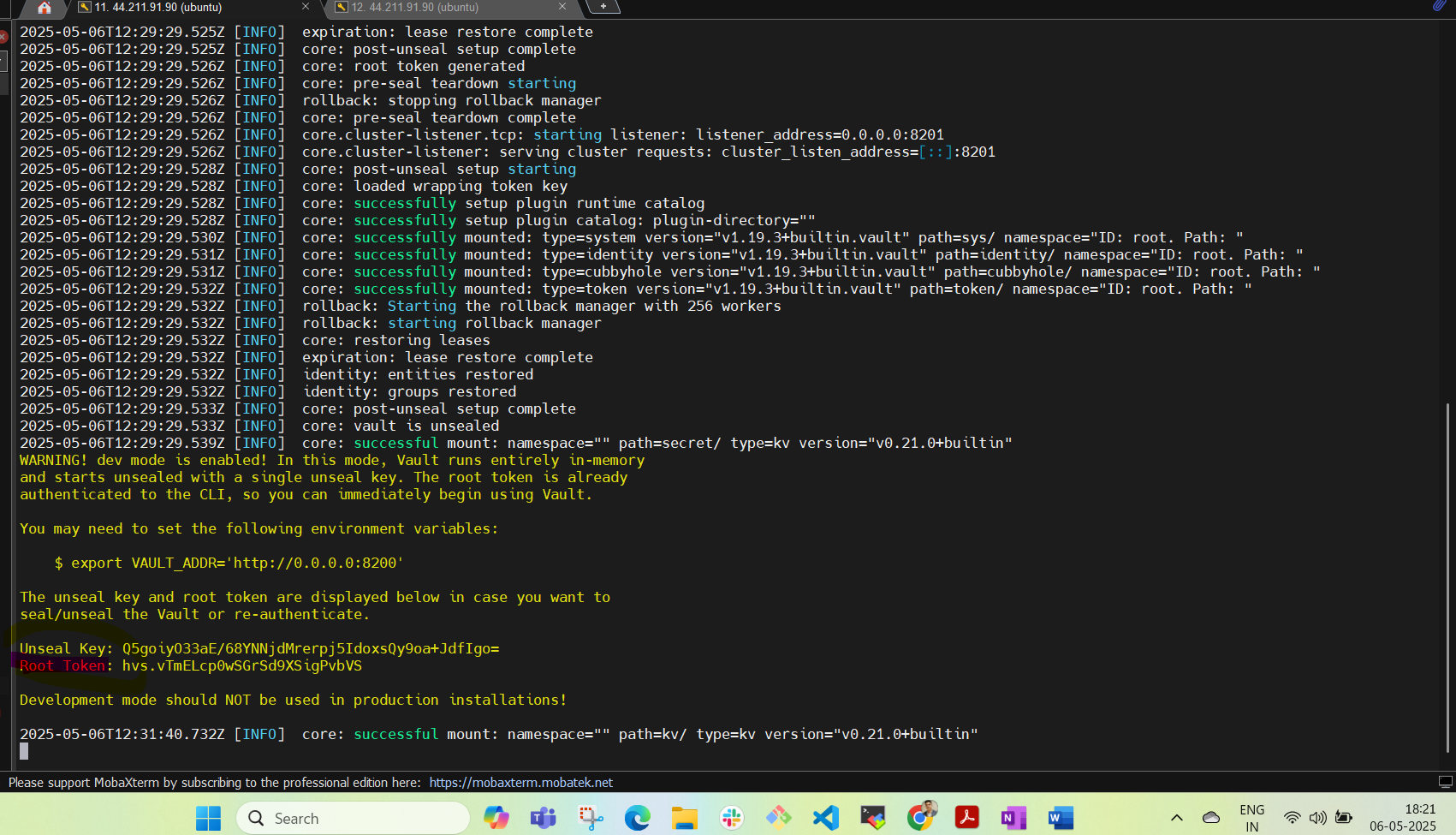
terraform apply

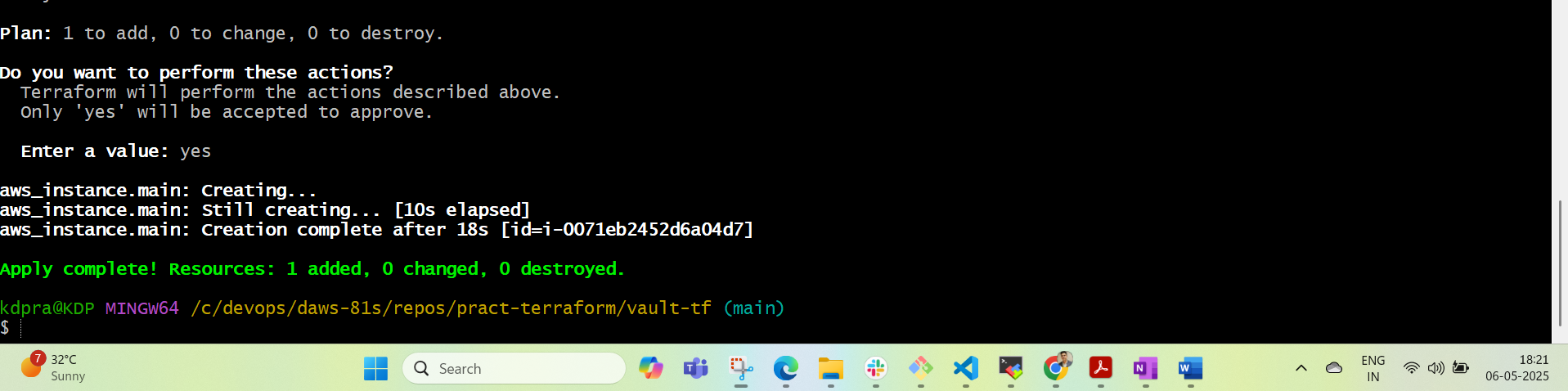
```



Role ID and Secret ID







Secret from Vault:

