Lesson 10 Demo 01

Managing Terraform State Using Default Local Backend

Objective: To illustrate the steps involved in managing Terraform state using the default local backend, making infrastructure changes, and verifying the updated state

Tools required: Visual Studio Code

Prerequisites: Ensure that you have created and implemented the AWS access key and secret key before starting this demo. Refer to Lesson 08 Assisted Practice 02 for detailed steps.

Note: The folder structure created in the previous demos is used here. It is also included in the resources section of LMS. Please refer Lesson_10_demo_01

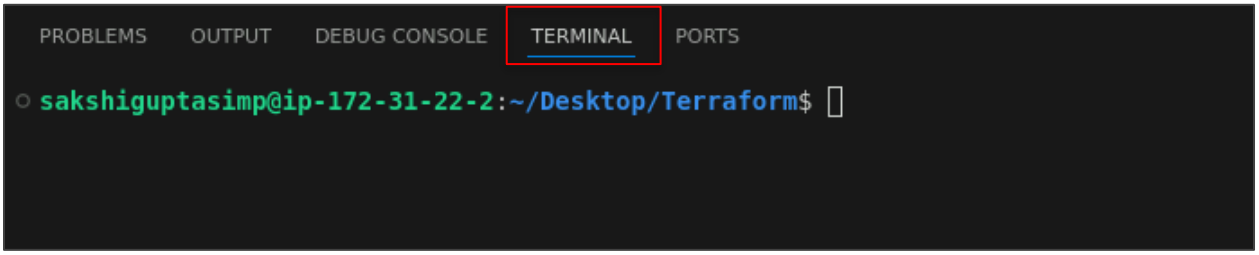
Steps to be followed:

1. Show current state
2. Show state file location
3. Modify, plan, and execute changes
4. Show new state and state backup

Note: Before starting the demo, download the folder structure provided in the LMS prerequisites. Make sure you have installed all the necessary modules.

Step 1: Show current state

1.1 Navigate to the **Terminal** in Visual Studio Code



1.2 Display the resources created and their details by running the following command:
terraform show

```
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform show
# aws_eip.nat_gateway_eip:
resource "aws_eip" "nat_gateway_eip" {
    allocation_id        = "eipalloc-0bfc6f1de31c4140e"
    association_id       = "eipassoc-01b08917f266b1c10"
    domain               = "vpc"
    id                   = "eipalloc-0bfc6f1de31c4140e"
    network_border_group = "us-east-1"
    network_interface    = "eni-07882c500eb445b7b"
    private_dns          = "ip-10-0-101-93.ec2.internal"
    private_ip           = "10.0.101.93"
    public_dns           = "ec2-100-24-227-134.compute-1.amazonaws.com"
    public_ip            = "100.24.227.134"
    public_ipv4_pool     = "amazon"
    tags                 = {
        "Name" = "demo_igw_eip"
    }
    tags_all             = {
        "Name" = "demo_igw_eip"
    }
    vpc                  = true
}

# aws_instance.ubuntu_server:
resource "aws_instance" "ubuntu_server" {
    ami                                  = "ami-0c819f65440d5f1d1"
    arn                                  = "arn:aws:ec2:us-east-1:089336525232:instance/i-0a364970543dc7ec4"
```

```
        http_put_response_hop_limit = 1
        http_tokens                 = "optional"
        instance_metadata_tags      = "disabled"
    }

    root_block_device {
        delete_on_termination = true
        device_name           = "/dev/sda1"
        encrypted             = false
        iops                  = 100
        tags                  = {}
        throughput            = 0
        volume_id             = "vol-0046405e603cfa450"
        volume_size           = 8
        volume_type           = "gp2"
    }
}


Outputs:

public_dns = "ec2-3-218-156-193.compute-1.amazonaws.com"
public_dns_server_subnet_1 = "ec2-54-242-253-155.compute-1.amazonaws.com"
public_ip = "3.218.156.193"
public_ip_server_subnet_1 = "54.242.253.155"
size = "t2.micro"
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$
```

1.3 List all items in the current Terraform state using the following command:
terraform state list

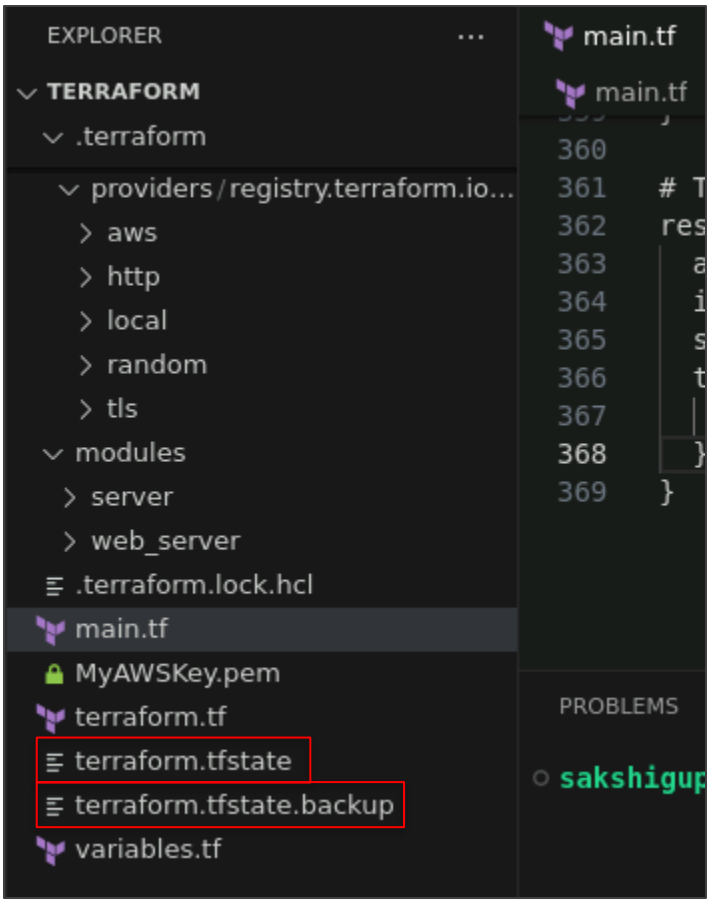
```
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform state list
data.aws_ami.ubuntu
data.aws_availability_zones.available
data.aws_region.current
aws_eip.nat_gateway_eip
aws_instance.ubuntu_server
aws_instance.web_server
aws_internet_gateway.internet_gateway
aws_key_pair.generated
aws_nat_gateway.nat_gateway
aws_route_table.private_route_table
aws_route_table.public_route_table
aws_route_table_association.private["private_subnet_1"]
aws_route_table_association.private["private_subnet_2"]
aws_route_table_association.private["private_subnet_3"]
aws_route_table_association.public["public_subnet_1"]
aws_route_table_association.public["public_subnet_2"]
aws_route_table_association.public["public_subnet_3"]
aws_security_group.ingress-ssh
aws_security_group.vpc-ping
aws_security_group.vpc-web
aws_subnet.private_subnets["private_subnet_1"]
aws_subnet.private_subnets["private_subnet_2"]
aws_subnet.private_subnets["private_subnet_3"]
aws_subnet.public_subnets["public_subnet_1"]
aws_subnet.public_subnets["public_subnet_2"]
```

```
aws_security_group.vpc-web
aws_subnet.private_subnets["private_subnet_1"]
aws_subnet.private_subnets["private_subnet_2"]
aws_subnet.private_subnets["private_subnet_3"]
aws_subnet.public_subnets["public_subnet_1"]
aws_subnet.public_subnets["public_subnet_2"]
aws_subnet.public_subnets["public_subnet_3"]
aws_vpc.vpc
local_file.private_key_pem
random_string.random
tls_private_key.generated
module.server.aws_instance.web
module.server_subnet_1.aws_instance.web
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$
```

This command provides a simple list of resource names that are part of the Terraform state, which can help the user quickly see what is being managed.

Step 2: Show state file location

2.1 By default, Terraform saves the state file as **terraform.tfstate** in the working directory. View the file named **terraform.tfstate** along with its backup **terraform.tfstate.backup** in Visual Studio Code:



terraform.tfstate


```
{
  "outputs": {
    "public_dns_server_subnet_1": {
      "value": "ec2-54-242-253-155.compute-1.amazonaws.com",
      "type": "string"
    },
    "public_ip": {
      "value": "3.218.156.193",
      "type": "string"
    },
    "public_ip_server_subnet_1": {
      "value": "54.242.253.155",
      "type": "string"
    },
    "size": {
      "value": "t2.micro",
      "type": "string"
```


main.tf | terraform.tfstate.backup × | variables.tf

terraform.tfstate.backup


```
{
  "version": 4,
  "terraform_version": "1.1.6",
  "serial": 30,
  "lineage": "e525d331-669b-5db4-02b1-df5cc4286ce6",
  "outputs": {
    "public_dns": {
      "value": "ec2-3-218-156-193.compute-1.amazonaws.com",
      "type": "string"
    },
    "public_dns_server_subnet_1": {
      "value": "ec2-54-242-253-155.compute-1.amazonaws.com",
      "type": "string"
    },
    "public_ip": {
      "value": "3.218.156.193",
      "type": "string"
```

Step 3: Modify, plan, and execute changes

3.1 Open the main Terraform configuration file, named as **main.tf**, in Visual Studio Code:

```
main.tf
/*
Name: IaC Buildout for Terraform Associate Exam
Description: AWS Infrastructure Buildout
Contributors: Bryan and Gabe
*/
variable "aws_access_key" {
  description = "AWS access key"
  type        = string
  default     = "ASIARJTG7GGYFAAT74ND"
}

variable "aws_secret_key" {
  description = "AWS secret key"
  type        = string
  default     = "JZI6zk4403bxXLW6cNgY80ctDO4E4XLPY54K/Xk5"
```

3.2 Add a new resource block to define an additional EC2 instance in the file main.tf using the following code:

```
resource "aws_instance" "web_server_2" {
  ami         = data.aws_ami.ubuntu.id
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.public_subnets["public_subnet_2"].id
  tags = {
    Name = "Web EC2 Server"
  }
}
```

```
# Terraform Resource Block - To Build EC2 instance in Public Subnet
resource "aws_instance" "web_server_2" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.public_subnets["public_subnet_2"].id
  tags = {
    Name = "Web EC2 Server"
  }
}
```

3.3 Ensure that the files are properly formatted and adhere to Terraform style conventions by using the following command:

terraform fmt

```
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform fmt
main.tf
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$
```

3.4 View the changes Terraform will make to the infrastructure based on the updated configuration using the following command:

terraform plan

```
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform plan
random_string.random: Refreshing state... [id=sN[Q3QR+lE]
tls_private_key.generated: Refreshing state... [id=6e0401af932d02263816c82fb45a6bf99c9fa346]
local_file.private_key_pem: Refreshing state... [id=1f539a7fa34d88b2e8a5cf0935e11ccb75d0ebf0]
aws_key_pair.generated: Refreshing state... [id=MyAWSKey]
aws_vpc.vpc: Refreshing state... [id=vpc-03ff423afc1e590f9]
aws_internet_gateway.internet_gateway: Refreshing state... [id=igw-0eca6af0c052ecceb]
aws_security_group.vpc-web: Refreshing state... [id=sg-0e769028febe6878e]
aws_security_group.ingress-ssh: Refreshing state... [id=sg-0576a04d9ef17e520]
aws_subnet.public_subnets["public_subnet_1"]: Refreshing state... [id=subnet-0c31bcb8850229119]
aws_subnet.private_subnets["private_subnet_2"]: Refreshing state... [id=subnet-04694bb5a1ab5f0bc]
aws_subnet.public_subnets["public_subnet_3"]: Refreshing state... [id=subnet-0a664a5f036daf138]
aws_security_group.vpc-ping: Refreshing state... [id=sg-033547863b787fd90]
aws_subnet.private_subnets["private_subnet_3"]: Refreshing state... [id=subnet-0727cef018cbf900c]
aws_subnet.private_subnets["private_subnet_1"]: Refreshing state... [id=subnet-0c5a2da8dd1781c27]
```

```
          + kms_key_id            = (known after apply)
          + tags                  = (known after apply)
          + throughput            = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }
    }

Plan: 1 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------------------------------------------

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if
you run "terraform apply" now.
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$
```

3.5 Apply these changes using the following command:

terraform apply

```
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform apply
tls_private_key.generated: Refreshing state... [id=6e0401af932d02263816c82fb45a6bf99c9fa346]
random_string.random: Refreshing state... [id=sN[Q3QR+lE]
local_file.private_key_pem: Refreshing state... [id=1f539a7fa34d88b2e8a5cf0935e11ccb75d0ebf0]
aws_key_pair.generated: Refreshing state... [id=MyAWSKey]
aws_vpc.vpc: Refreshing state... [id=vpc-03ff423afc1e590f9]
aws_internet_gateway.internet_gateway: Refreshing state... [id=igw-0eca6af0c052ecceb]
aws_security_group.ingress-ssh: Refreshing state... [id=sg-0576a04d9ef17e520]
aws_subnet.private_subnets["private_subnet_2"]: Refreshing state... [id=subnet-04694bb5a1ab5f0bc]
aws_subnet.private_subnets["private_subnet_3"]: Refreshing state... [id=subnet-0727cef018cbf900c]
aws_subnet.public_subnets["public_subnet_3"]: Refreshing state... [id=subnet-0a664a5f036daf138]
aws_subnet.private_subnets["private_subnet_1"]: Refreshing state... [id=subnet-0c5a2da8dd1781c27]
aws_subnet.public_subnets["public_subnet_2"]: Refreshing state... [id=subnet-0d7efc435ad6e3ab5]
aws_subnet.public_subnets["public_subnet_1"]: Refreshing state... [id=subnet-0c31bcb8850229119]
aws_security_group.vpc-ping: Refreshing state... [id=sg-033547863b787fd90]
```

3.6 Confirm by typing **yes** when prompted

```
          + kms_key_id            = (known after apply)
          + tags                  = (known after apply)
          + throughput            = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

Step 4: Show new state and state backup

4.1 After applying the changes, verify that the new EC2 instance is included in the managed state using the following command:
terraform state list

```
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform state list
data.aws_ami.ubuntu
data.aws_availability_zones.available
data.aws_region.current
aws_eip.nat_gateway_eip
aws_instance.ubuntu_server
```

```
aws_instance.ubuntu_server
aws_instance.web_server
aws_instance.web_server_2
aws_internet_gateway.internet_gateway
aws_key_pair.generated
aws_nat_gateway.nat_gateway
aws_route_table.private_route_table
aws_route_table.public_route_table
aws_route_table_association.private["private_subnet_1"]
aws_route_table_association.private["private_subnet_2"]
aws_route_table_association.private["private_subnet_3"]
aws_route_table_association.public["public_subnet_1"]
aws_route_table_association.public["public_subnet_2"]
aws_route_table_association.public["public_subnet_3"]
aws_security_group.ingress-ssh
aws_security_group.vpc-ping
```

By following these steps, you have successfully managed Terraform state using the default local backend.