Lab: Terraform State Command

The `terraform state` command is used for advanced state management. As your Terraform usage becomes more advanced, there are some cases where you may need to modify the Terraform state. Rather than modify the state directly, the `terraform state` commands can be used in many cases instead.

- Task 1: Deploy Infrastructure Using Terraform
- Task 2: Utilize the `terraform show` command to show state information
- Task 3: Utilize the `terraform state` command to show state information
- Task 4: Utilize the `terraform state` command to list resource information
- Task 5: Utilize the `terraform state` command to show resource information

Task 1: Deploy Infrastructure Using Terraform

If you have not already deployed your infrastructure from a previous lab create the following Terraform configuration files.

`terraform.tf`

```
terraform {
  required_version = ">= 1.0.0"
  required_providers {
    aws = {
      source = "hashicorp/aws"
    }
    http = {
      source  = "hashicorp/http"
      version = "2.1.0"
    }
    random = {
      source  = "hashicorp/random"
      version = "3.0.0"
    }
    local = {
      source  = "hashicorp/local"
      version = "2.1.0"
    }
    tls = {
      source  = "hashicorp/tls"
      version = "3.1.0"
    }
  }
```


```
}
```

main.tf

```
/*
Name: IaC Buildout for Terraform Associate Exam
Description: AWS Infrastructure Buildout
Contributors: Bryan and Gabe
*/

provider "aws" {
  region = "us-east-1"
  default_tags {
    tags = {
      Environment = terraform.workspace
      Owner       = "TF Hands On Lab"
      Project     = "Infrastructure as Code"
      Terraform   = "true"
    }
  }
}


#Retrieve the list of AZs in the current AWS region
data "aws_availability_zones" "available" {}
data "aws_region" "current" {}

#Define the VPC
resource "aws_vpc" "vpc" {
  cidr_block = var.vpc_cidr

  tags = {
    Name        = var.vpc_name
    Environment = "demo_environment"
    Terraform   = "true"
  }
}

#Deploy the private subnets
resource "aws_subnet" "private_subnets" {
  for_each          = var.private_subnets
  vpc_id            = aws_vpc.vpc.id
  cidr_block        = cidrsubnet(var.vpc_cidr, 8, each.value)
  availability_zone = tolist(data.aws_availability_zones.available.names)[each.value]

  tags = {
    Name      = each.key
    Terraform = "true"
  }
```


```
}

#Deploy the public subnets
resource "aws_subnet" "public_subnets" {
  for_each                = var.public_subnets
  vpc_id                  = aws_vpc.vpc.id
  cidr_block              = cidrsubnet(var.vpc_cidr, 8, each.value + 100)
  availability_zone       = tolist(data.aws_availability_zones.available.names)[each.val
  map_public_ip_on_launch = true

  tags = {
    Name      = each.key
    Terraform = "true"
  }
}

#Create route tables for public and private subnets
resource "aws_route_table" "public_route_table" {
  vpc_id = aws_vpc.vpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.internet_gateway.id
    #nat_gateway_id = aws_nat_gateway.nat_gateway.id
  }
  tags = {
    Name      = "demo_public_rtb"
    Terraform = "true"
  }
}

resource "aws_route_table" "private_route_table" {
  vpc_id = aws_vpc.vpc.id

  route {
    cidr_block = "0.0.0.0/0"
    # gateway_id     = aws_internet_gateway.internet_gateway.id
    nat_gateway_id = aws_nat_gateway.nat_gateway.id
  }
  tags = {
    Name      = "demo_private_rtb"
    Terraform = "true"
  }
}

#Create route table associations
resource "aws_route_table_association" "public" {
  depends_on     = [aws_subnet.public_subnets]
  route_table_id = aws_route_table.public_route_table.id
```


```
  for_each       = aws_subnet.public_subnets
  subnet_id      = each.value.id
}

resource "aws_route_table_association" "private" {
  depends_on     = [aws_subnet.private_subnets]
  route_table_id = aws_route_table.private_route_table.id
  for_each       = aws_subnet.private_subnets
  subnet_id      = each.value.id
}

#Create Internet Gateway
resource "aws_internet_gateway" "internet_gateway" {
  vpc_id = aws_vpc.vpc.id
  tags = {
    Name = "demo_igw"
  }
}

#Create EIP for NAT Gateway
resource "aws_eip" "nat_gateway_eip" {
  vpc        = true
  depends_on = [aws_internet_gateway.internet_gateway]
  tags = {
    Name = "demo_igw_eip"
  }
}

#Create NAT Gateway
resource "aws_nat_gateway" "nat_gateway" {
  depends_on    = [aws_subnet.public_subnets]
  allocation_id = aws_eip.nat_gateway_eip.id
  subnet_id     = aws_subnet.public_subnets["public_subnet_1"].id
  tags = {
    Name = "demo_nat_gateway"
  }
}

# Terraform Data Block - To Lookup Latest Ubuntu 20.04 AMI Image
data "aws_ami" "ubuntu" {
  most_recent = true

  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*"]
  }

  filter {
    name   = "virtualization-type"
```


```
    values = ["hvm"]
  }

  owners = ["099720109477"]
}

# Terraform Resource Block - To Build EC2 instance in Public Subnet
resource "aws_instance" "web_server" {
  ami              = data.aws_ami.ubuntu.id
  instance_type    = "t2.micro"
  subnet_id        = aws_subnet.public_subnets["public_subnet_1"].id
  security_groups = [aws_security_group.vpc-ping.id, aws_security_group.ingress-ssh.id,
  key_name         = aws_key_pair.generated.key_name
  connection {
    user        = "ubuntu"
    private_key = tls_private_key.generated.private_key_pem
    host        = self.public_ip
  }
  associate_public_ip_address = true
  tags = {
    Name = "Web EC2 Server"
  }

  provisioner "local-exec" {
    command = "chmod 600 ${local_file.private_key_pem.filename}"
  }

  provisioner "remote-exec" {
    inline = [
      "git clone https://github.com/hashicorp/demo-terraform-101",
      "cp -a demo-terraform-101/. /tmp/",
      "sudo sh /tmp/assets/setup-web.sh",
    ]
  }
}

# Terraform Resource Block - Security Group to Allow Ping Traffic
resource "aws_security_group" "vpc-ping" {
  name        = "vpc-ping"
  vpc_id      = aws_vpc.vpc.id
  description = "ICMP for Ping Access"
  ingress {
    description = "Allow ICMP Traffic"
    from_port   = -1
    to_port     = -1
    protocol    = "icmp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
```


```
    description = "Allow all ip and ports outboun"
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

# Terraform Data Block - Lookup Ubuntu 16.04
data "aws_ami" "ubuntu_16_04" {
  most_recent = true

  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-*"]
  }

  owners = ["099720109477"]
}

data "aws_ami" "windows_2019" {
  most_recent = true
  filter {
    name   = "name"
    values = ["Windows_Server-2019-English-Full-Base-*"]
  }
  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
  owners = ["801119661308"] # Canonical
}

resource "tls_private_key" "generated" {
  algorithm = "RSA"
}

resource "local_file" "private_key_pem" {
  content  = tls_private_key.generated.private_key_pem
  filename = "MyAWSKey.pem"
}

resource "aws_key_pair" "generated" {
  key_name   = "MyAWSKey"
  public_key = tls_private_key.generated.public_key_openssh

  lifecycle {
    ignore_changes = [key_name]
  }
```


```
}

# Security Groups

resource "aws_security_group" "ingress-ssh" {
  name   = "allow-all-ssh"
  vpc_id = aws_vpc.vpc.id
  ingress {
    cidr_blocks = [
      "0.0.0.0/0"
    ]
    from_port = 22
    to_port   = 22
    protocol  = "tcp"
  }
  // Terraform removes the default rule
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

# Create Security Group - Web Traffic
resource "aws_security_group" "vpc-web" {
  name        = "vpc-web-${terraform.workspace}"
  vpc_id      = aws_vpc.vpc.id
  description = "Web Traffic"
  ingress {
    description = "Allow Port 80"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    description = "Allow Port 443"
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    description = "Allow all ip and ports outbound"
    from_port   = 0
    to_port     = 0
```


```
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

variables.tf

```
variable "aws_region" {
  type    = string
  default = "us-east-1"
}

variable "vpc_name" {
  type    = string
  default = "demo_vpc"
}

variable "vpc_cidr" {
  type    = string
  default = "10.0.0.0/16"
}

variable "private_subnets" {
  default = {
    "private_subnet_1" = 1
    "private_subnet_2" = 2
    "private_subnet_3" = 3
  }
}

variable "public_subnets" {
  default = {
    "public_subnet_1" = 1
    "public_subnet_2" = 2
    "public_subnet_3" = 3
  }
}
```

Save your files and issue a `terrraform init` and `terraform apply` to build out the infrastructure.

```
terraform init
terraform apply
```

```
Plan: 25 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
```

Task 2: Utilize the `terraform show` command to show state information

You can issue a `terraform show` to see the resources that were created by Terraform.

```
terraform show
```

```
# aws_eip.nat_gateway_eip:
resource "aws_eip" "nat_gateway_eip" {
    domain               = "vpc"
    id                   = "eipalloc-0260c99a3a7a12677"
    network_border_group = "us-east-1"
    public_dns           = "ec2-3-92-117-57.compute-1.amazonaws.com"
    public_ip            = "3.92.117.57"
    public_ipv4_pool     = "amazon"

...Redacted for brevity...

# tls_private_key.generated:
resource "tls_private_key" "generated" {
    algorithm                   = "RSA"
    ecdsa_curve                 = "P224"
    id                          = "999595a596d4b949afefcc27746a9c32b3582667"
    private_key_pem             = (sensitive value)
    public_key_fingerprint_md5 = "8e:c1:c9:5c:05:85:62:b4:bb:fc:2e:45:5a:b9:93:4a"
    public_key_openssh          = <<-EOT
        ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDLwvogPirs6RgzNkh68RK+Nq//Pqi1fucmSQfzv2N7
    EOT
    public_key_pem              = <<-EOT
        -----BEGIN PUBLIC KEY-----
        MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAy8L6ID4q7OkYMzZIevES
        vjav/z6otX7nJkkH879je9aH82sZeComjWmws1qlX1FUGRlKCdmKXvC779N/N0FJ
        YAWcWUIyDGwXALK+ZeratVRmdBDetrh4vXWHEtVH/oxJBWNbT0HYTeN7S65UbDIW
        irWzN2olzGjSXRjZRCLzbLvePPsp+ETrZ1Jr6b5iprNeO0QE6BJv9mMNlSvhYtwv
        wVy5Qb+wNPtYvmTLcXEUNRgk6r3pddk7FsXLT3EfS888XgwR/AJwI3wF39ROHi76
        gC7hOwKf8kb9IPs5+tHdcuw7TduRPawutKL1fBBVoI7lgVT+uCUg8N7+tARrJ+hV
        NwIDAQAB
        -----END PUBLIC KEY-----
    EOT
    rsa_bits                    = 2048
}
```

You will note that there is a lot of information Terraform stores about all the resources it manages within

it's state file. To view and manage these resources in an easier way we will use the `terraform state` command.

Task 3: Utilize the `terraform state` command to show state information

You can issue a `terraform state` to see the options for performing more granualar operations related to the resources stored within Terraform state.

```
terraform state
```

```
Usage: terraform [global options] state <subcommand> [options] [args]

  This command has subcommands for advanced state management.

  These subcommands can be used to slice and dice the Terraform state.
  This is sometimes necessary in advanced cases. For your safety, all
  state management commands that modify the state create a timestamped
  backup of the state prior to making modifications.

  The structure and output of the commands is specifically tailored to work
  well with the common Unix utilities such as grep, awk, etc. We recommend
  using those tools to perform more advanced state tasks.

Subcommands:
    list                List resources in the state
    mv                  Move an item in the state
    pull                Pull current state and output to stdout
    push                Update remote state from a local state file
    replace-provider    Replace provider in the state
    rm                  Remove instances from the state
    show                Show a resource in the state
```

Task 4: Utilize the `terraform state` command to list resource information

You can issue a `terraform state list` in either workspace to see the resources that each of the workspaces is managing.

```
terraform state list
```

```
data.aws_ami.ubuntu
data.aws_ami.ubuntu_16_04
data.aws_ami.windows_2019
data.aws_availability_zones.available
```


```
data.aws_region.current
aws_eip.nat_gateway_eip
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_subnet.public_subnets["public_subnet_3"]
aws_vpc.vpc
local_file.private_key_pem
tls_private_key.generated
```

Task 5: Utilize the `terraform state` command to show resource information

You can issue a `terraform state show` in either workspace to see the resources that each of the workspaces is managing.

```
terraform state show aws_instance.web_server
```

```
# aws_instance.web_server:
resource "aws_instance" "web_server" {
    ami                                  = "ami-083654bd07b5da81d"
    arn                                  = "arn:aws:ec2:us-east-1:508140242758:instance/
    associate_public_ip_address          = true
    availability_zone                    = "us-east-1b"
    cpu_core_count                       = 1
    cpu_threads_per_core                 = 1
    disable_api_termination              = false
    ebs_optimized                        = false
    get_password_data                    = false
    hibernation                          = false
```


```
    id                                     = "i-0f87913a4b4da9db5"
    instance_initiated_shutdown_behavior = "stop"
    instance_state                         = "running"
    instance_type                          = "t2.micro"
    ipv6_address_count                     = 0
    ipv6_addresses                         = []
    key_name                               = "MyAWSKey"
    monitoring                             = false
    primary_network_interface_id           = "eni-04039cf0944805906"
    private_dns                            = "ip-10-0-101-91.ec2.internal"
    private_ip                             = "10.0.101.91"
    public_ip                              = "3.236.193.131"
    secondary_private_ips                  = []
    security_groups                        = [
        "sg-003059856f83334c0",
        "sg-0986589e4d7cdc719",
        "sg-0b9c402b6331dcb1e",
    ]
    source_dest_check                      = true
    subnet_id                              = "subnet-0ea34d249acab1fdb"
    tags                                   = {
        "Name" = "Web EC2 Server"
    }
    tags_all                               = {
        "Environment" = "default"
        "Name"        = "Web EC2 Server"
        "Owner"       = "TF Hands On Lab"
        "Project"     = "Infrastructure as Code"
        "Terraform"   = "true"
    }
    tenancy                                = "default"
    vpc_security_group_ids                 = [
        "sg-003059856f83334c0",
        "sg-0986589e4d7cdc719",
        "sg-0b9c402b6331dcb1e",
    ]

    capacity_reservation_specification {
        capacity_reservation_preference = "open"
    }

    credit_specification {
        cpu_credits = "standard"
    }

    enclave_options {
        enabled = false
    }
```


```
    metadata_options {
        http_endpoint               = "enabled"
        http_put_response_hop_limit = 1
        http_tokens                 = "optional"
    }

    root_block_device {
        delete_on_termination = true
        device_name           = "/dev/sda1"
        encrypted             = false
        iops                  = 100
        tags                  = {}
        throughput            = 0
        volume_id             = "vol-04c74f0003ac59c4c"
        volume_size           = 8
        volume_type           = "gp2"
    }
}
```

The `terraform state` command can also be used to manipulate state information. Rather than modify the state directly, the `terraform state` commands can be used in many cases instead. This includes moving, replacing, and overwriting resources and state information.

> Note: Modifying state information directly is typically not required and should only be used in advanced cases. For your safety, all state management commands that modify the state create a timestamped backup of the state prior to making modifications. Exercise caution.

The output and command-line structure of the state subcommands is designed to be usable with Unix command-line tools such as grep, awk, and similar PowerShell commands.

Reference

Terraform State Command