Day-01

Cloud Network Tools :

AWS: CFT

Azure: ARM

GCP: Deployment Manager

Disadvantages of **CFT and ARM:**

- JSON or YAML All the config should be dumped in the same file it self only. like s3,subnets all resources in one file
- Learning JSON ,YAML Bit complex.
- Importing of resource is Complex in AWS and this Option Not there in Azure.
- CFT will be working with in AWS & ARM In Azure only.
- Module Concepts not there in Azure or AWS.
- Workspace Option Not Available in Azure and AWS
- Performing dry checks.

Hashicorp:

- Terraform : IAC automation
- Packer : Image Automation
- consul : cluster & service Discovery
- vault : store secrets
- nomad

Terraform Commands:

```
terraform init
terraform fmt
terraform validate
terraform plan
terraform apply
terraform destroy
terraform state list → What Resource is Created
```

We use this Based on Requirement in Different Environments

```
# We strongly recommend using the required_providers block to set the
# Azure Provider source and version being used
terraform {
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "=3.0.0"
    }
  }
}
```

main.tf

```
# cloud Provider
provider "aws" {
  region = "us-east-1"

}

# Creating VPC
resource "aws_vpc" "day1-vpc" {
  cidr_block           = "10.0.0.0/16"
```

```
  enable_dns_hostnames = "true"

  tags = {
    Name = "day1-vpc"
  }

}



#IGW
resource "aws_internet_gateway" "day1-Igw" {
  vpc_id = aws_vpc.day1-vpc.id

  tags = {
    Name = "day1-Igw"
  }

}

#Subnets
resource "aws_subnet" "day1-subnet-1" {
  vpc_id     = aws_vpc.day1-vpc.id
  cidr_block = "10.0.1.0/24"

  tags = {
    Name = "day1-subnet-1"
  }

}

#RouteTable
resource "aws_route_table" "day1-RouteTable" {
  vpc_id = aws_vpc.day1-vpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.day1-Igw.id
  }
```

```
  tags = {
    Name    = "day1-RouteTable"
    Service = "Terraform"
  }
}


#RouteTable Association
resource "aws_route_table_association" "day1-RTA" {
  subnet_id      = aws_subnet.day1-subnet-1.id
  route_table_id = aws_route_table.day1-RouteTable.id
}


#Security Groups
resource "aws_security_group" "day1-SG" {
  name        = "day1-SG"
  description = "Allow TLS inbound traffic and all outbound traffic"
  vpc_id      = aws_vpc.day1-vpc.id

  ingress {
    from_port   = 0
    protocol    = "-1"
    to_port     = 0
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    protocol    = "-1"
    to_port     = 0
    cidr_blocks = ["0.0.0.0/0"]
  }
```

```
  tags = {
    Name = "day1-SG"
  }
}
```

Data Sources:

We created VPC Through Terraform and Attach IGW Through Terraform , What If VPC is Created Manually Through Console We Have to Attach IGW to VPC by using Datasources We can do that.

datasource.tf

```
# TO get the Manually Created VPC
data "aws_vpc" "datasource-vpc" {
  id = "vpc-0c036eeba55cec6af"
}


#we are attaching IGW manualluy created VPC  check vpc_id

resource "aws_internet_gateway" "datasource-igw" {
  vpc_id = data.aws_vpc.datasource-vpc.id

  tags = {
    Name="datasource-igw"
  }
}
```

Terraform Datasources along with Remote State:

Suppose We Create a Base Infra Like Vpc, igw, subnet, RT,RTA Like that we Store the state file in S3 Bucket , By Using That State file We can Deploy EC2 separately. We can separate state file in S3 For ec2 instance.

BaseInfra:

main.tf

```
#This Terraform Code Deploys Basic VPC Infra.
provider "aws" {

  region = var.aws_region
}

terraform {
  backend "s3" {
    bucket = "terraform250304"
    key    = "base-infra.tfstate"
    region = "us-east-1"

  }
}

resource "aws_vpc" "default" {
  cidr_block           = var.vpc_cidr
  enable_dns_hostnames = true
  tags = {
    Name  = "${var.vpc_name}"
    Owner = "Saikiran"
  }
}

resource "aws_internet_gateway" "default" {
  vpc_id = aws_vpc.default.id
  tags = {
    Name = "${var.IGW_name}"
  }
}
```

```
resource "aws_subnet" "subnet1-public" {
  vpc_id            = aws_vpc.default.id
  cidr_block        = var.public_subnet1_cidr
  availability_zone = "us-east-1a"

  tags = {
    Name = "${var.public_subnet1_name}"
  }
}

resource "aws_subnet" "subnet2-public" {
  vpc_id            = aws_vpc.default.id
  cidr_block        = var.public_subnet2_cidr
  availability_zone = "us-east-1b"

  tags = {
    Name = "${var.public_subnet2_name}"
  }
}

resource "aws_subnet" "subnet3-public" {
  vpc_id            = aws_vpc.default.id
  cidr_block        = var.public_subnet3_cidr
  availability_zone = "us-east-1c"

  tags = {
    Name = "${var.public_subnet3_name}"
  }

}


resource "aws_route_table" "terraform-public" {
  vpc_id = aws_vpc.default.id

  route {
    cidr_block = "0.0.0.0/0"
```

```
    gateway_id = aws_internet_gateway.default.id
  }

  tags = {
    Name = "${var.Main_Routing_Table}"
  }
}

resource "aws_route_table_association" "terraform-public" {
  subnet_id      = aws_subnet.subnet1-public.id
  route_table_id = aws_route_table.terraform-public.id
}

resource "aws_security_group" "allow_all" {
  name        = "allow_all"
  description = "Allow all inbound traffic"
  vpc_id      = aws_vpc.default.id

  ingress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

variable.tf

```
# variable "aws_access_key" {}
# variable "aws_secret_key" {}
variable "aws_region" {}
variable "amis" {
   description = "AMIs by region"
   default = {
      us-east-1 = "ami-97785bed" # ubuntu 14.04 LTS
      us-east-2 = "ami-f63b1193" # ubuntu 14.04 LTS
      us-west-1 = "ami-824c4ee2" # ubuntu 14.04 LTS
      us-west-2 = "ami-f2d3638a" # ubuntu 14.04 LTS
   }
}
variable "vpc_cidr" {}
variable "vpc_name" {}
variable "IGW_name" {}
variable "key_name" {}
variable "public_subnet1_cidr" {}
variable "public_subnet2_cidr" {}
variable "public_subnet3_cidr" {}
# variable "private_subnet_cidr" {}
variable "public_subnet1_name" {}
variable "public_subnet2_name" {}
variable "public_subnet3_name" {}
# variable "private_subnet_name" {}
variable Main_Routing_Table {}
variable "azs" {
 description = "Run the EC2 Instances in these Availability Zones"
 type = list(string)
 default = ["us-east-1a", "us-east-1b", "us-east-1c"]
}
variable "environment" { default = "dev" }
variable "instance_type" {
 type = map(string)
 default = {
  dev = "t2.nano"
  test = "t2.micro"
  prod = "t2.medium"
  }
```

```
}
```

terraform.tfvars

```
# aws_access_key = "xxxxxx"
# aws_secret_key = "yyyyyyy"
aws_region         = "us-east-1"
vpc_cidr           = "10.1.0.0/16"
public_subnet1_cidr = "10.1.1.0/24"
public_subnet2_cidr = "10.1.2.0/24"
public_subnet3_cidr = "10.1.3.0/24"
# private_subnet_cidr = "10.1.20.0/24"
vpc_name           = "terraform-aws-testing"
IGW_name           = "terraform-aws-igw"
public_subnet1_name = "Terraform_Public_Subnet1-testing"
public_subnet2_name = "Terraform_Public_Subnet2-testing"
public_subnet3_name = "Terraform_Public_Subnet3-testing"
# private_subnet_name = "Terraform_Private_Subnet-testing"
Main_Routing_Table = "Terraform_Main_table-testing"
key_name          = "aws"
environment       = "dev"
```

By Using Above one We can Create A Basic Infra , Now We have To add a resource EC2

main.tf

```
resource "aws_instance" "web-1" {
   ami = "ami-04b4f1a9cf54c11d0"
   availability_zone = "us-east-1a"
   instance_type = "t2.micro"
   key_name = "aws"
   subnet_id = data.aws_subnet.Terraform_Public_Subnet1-testing.id
```

```
    vpc_security_group_ids = [data.aws_security_group.allow_all.id]
    associate_public_ip_address = true
    tags = {
        Name = "Server-1"
        Env = "Prod"
        Owner = "Teja"
          CostCenter = "ABCD"

}
}



terraform {
  backend "s3" {
    bucket = "terraform250304"
    key = "current-state.tfstate"
    region = "us-east-1"

  }
}
```

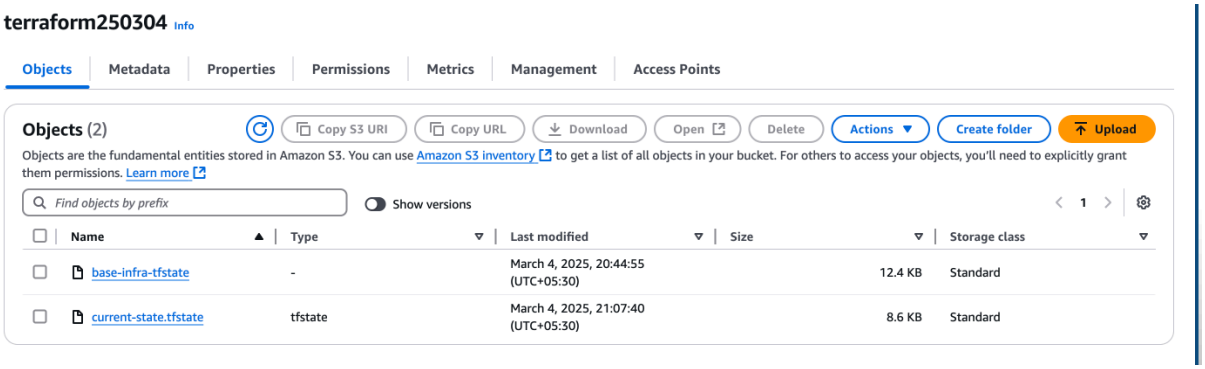
data-source.tf

```
data "aws_vpc" "terraform-aws-testing" {
  id = "vpc-080c60024c21fda34"
 }

data "aws_subnet" "Terraform_Public_Subnet1-testing" {
  id = "subnet-08e369e685a3c92db"
}

data "aws_security_group" "allow_all"{
  id = "sg-0dcea5f0cfa781c84"
}
```

Like these State Files Created


terraform250304 Info
Objects
Metadata
Properties
Permissions
Metrics
Management
Access Points
Objects (2)
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Show versions
1
Name
Type
Last modified
Size
Storage class
base-infra-tfstate
-
March 4, 2025, 20:44:55 (UTC+05:30)
12.4 KB
Standard
current-state.tfstate
tfstate
March 4, 2025, 21:07:40 (UTC+05:30)
8.6 KB
Standard