Terraform block will have terraform version, required_providers and backend configuration

Arguments are defined while creating resources and attributes are the resource details after creating the resource, the defined arguments will also become attributes of that resource after apply. Like ami, instance type for ec2.
Attribute reference is when we use an attribute using aws_instance.name.instance_type

On running terraform init .terraform directory and .terraform.lock.hcl file will be created.
.terraform directory will have provider plug-ins. Lock file will have provider version, constraints and hashes

On running terraform apply for the first time and once the resource creation is successful state file(terraform.tfstate) will be created in local which has the Metadata of all terraform managed infrastructure.

List and Map datatype
List: to pass multiple input variables
type = list(string)
default = [val1,val2,val3]
Reference:var.listname[0]
Map: to have a key value type of input variables
type = map(string)
default = {
key1=val1
key2=val2
}
Reference: var.mapname[key1]

Output with splat operator
output “public-ip-list-splat” {
value = aws_instance.myinstances[*].public_ip
}

for_each meta argument will accept a map or a set of string and creates an instance for each item in that map or set
Ex: resource aws_instance{
for_each = (str1, str2, str3)
tags {Name = “myec2-${each.key}”}
}

terraform_remote_state data source is used to fetch details of resources created in a different project

terraform-project/

```
|--- versions.tf
├── main.tf
├── variables.tf
├── outputs.tf
├── backend.tf
├── dev.tfvars
├── test.tfvars
├──
├── modules/
│   ├── vpc/
│   │   └── main.tf
│   ├── ec2/
│   │   └── main.tf
│   └── security-group/
│       └── main.tf
```

```
#####################################################
# Versions.tf
#####################################################
terraform {
  required_version = ">= 1.3.0, < 1.5.0"

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 5.0, < 5.10"
    }
  }

  backend "s3" {
    bucket         = "my-terraform-state-bucket"
    key            = "infra/terraform.tfstate"
    region         = "ap-south-1"
    dynamodb_table = "terraform-lock-table"
    encrypt        = true
  }
}
```

Or

```
terraform {
  backend "s3" {
    bucket       = "my-terraform-state-bucket"
```

```
    key          = "path/to/my/key"
    region       = "us-east-1"
    encrypt      = true
    use_lockfile = true
  }
}


#####################################################
# Provider.tf
#####################################################
provider "aws" {
  alias = “india”
  region = ap-south1
  profile = “default” #aws profile configured under aws configure
}
provider “aws” {
  alias = “us”
  region = “us-east1
}
provider “azurerm“{
  alias = “azure”
  region = “az-reg”
}
#####################################################
# Variables.tf
#####################################################

variable "vpc_cidr" {}
variable "public_subnet_cidr" {}
variable "instance_type" {}
variable "key_name" {}
variable "ami_id" {}
variable "s3_bucket_name" {}
variable "iam_user_name" {}
variable "iam_policy_name" {}

#####################################################
# Main.tf (root)
#####################################################
module "vpc" {
  source = "./modules/vpc"
  vpc_cidr           = var.vpc_cidr
  public_subnet_cidr = var.public_subnet_cidr
```

```
  tags              = { Project = "modular-tf" }
}

module "sg" {
  source = "./modules/security-group"
  vpc_id = module.vpc.vpc_id
  tags   = { Project = "modular-tf" }
}

module "ec2" {
  source = "./modules/ec2"
  ami_id        = var.ami_id
  instance_type = var.instance_type
  key_name      = var.key_name
  subnet_id     = module.vpc.public_subnet_id
  sg_id         = module.sg.sg_id
  tags          = { Project = "modular-tf" }
  depends_on    = [module.sg]
}

module "s3" {
  source = "./modules/s3"
  bucket_name = var.s3_bucket_name
  tags        = { Project = "modular-tf" }
}

module "iam" {
  source      = "./modules/iam"
  user_name   = var.iam_user_name
  policy_name = var.iam_policy_name
}

#####################################################
# Outputs.tf
#####################################################
output "vpc_id" {
  value = module.vpc.vpc_id
}

output "ec2_public_ip" {
  value = module.ec2.public_ip
}

output "s3_bucket" {
```

```
  value = module.s3.bucket_id
}

output "iam_user_arn" {
  value = module.iam.user_arn
}

#####################################################
# modules/vpc/main.tf
#####################################################
# aws_vpc is resource type
# main is resource local name
#cidr_block is argument
resource "aws_vpc" "main" {
  cidr_block           = var.vpc_cidr
  enable_dns_hostnames = true

  lifecycle {
    create_before_destroy = true
  }

  tags = merge(var.tags, { Name = "vpc" })
}

resource "aws_subnet" "public" {
  vpc_id                  = aws_vpc.main.id
  cidr_block              = var.public_subnet_cidr
  map_public_ip_on_launch = true

  tags = merge(var.tags, { Name = "public-subnet" })
}

output "vpc_id" {
  value = aws_vpc.main.id
}

output "public_subnet_id" {
  value = aws_subnet.public.id
}

#####################################################
# modules/security-group/main.tf
#####################################################
resource "aws_security_group" "web_sg" {
```

```
  name        = "web-sg"
  description = "Allow SSH/HTTP"
  vpc_id      = var.vpc_id

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

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

  lifecycle {
    prevent_destroy = false
  }

  tags = merge(var.tags, { Name = "web-sg" })
}

output "sg_id" {
  value = aws_security_group.web_sg.id
}

#####################################################
# modules/ec2/main.tf
#####################################################
resource "aws_instance" "web" {
  ami                    = var.ami_id
  instance_type          = var.instance_type
  key_name               = var.key_name
  subnet_id              = var.subnet_id
```

```
  vpc_security_group_ids = [var.sg_id]
  associate_public_ip_address = true
  count = 1

  tags = merge(var.tags, { Name = "web-ec2-${count.index}" })

  lifecycle {
    create_before_destroy = true
    ignore_changes        = [tags]
  }

  # Provisioners
  provisioner "file" {
    source      = "app/index.html"
    destination = "/tmp/index.html"

    connection {
      type        = "ssh"
      user        = "ec2-user"
      private_key = file("~/.ssh/${var.key_name}.pem")
      host        = self.public_ip
    }
  }

  provisioner "local-exec" {
    command = "echo 'EC2 instance created: ${self.id}' >> instance.log"
  }

  provisioner "remote-exec" {
    inline = [
      "sudo yum update -y",
      "sudo yum install -y httpd",
      "sudo systemctl start httpd",
      "sudo mv /tmp/index.html /var/www/html/index.html"
    ]

    connection {
      type        = "ssh"
      user        = "ec2-user"
      private_key = file("~/.ssh/${var.key_name}.pem")
      host        = self.public_ip
    }
  }
}
```

```
output "public_ip" {
  value = aws_instance.web.*.public_ip
}

#####################################################
# modules/s3/main.tf
#####################################################
resource "aws_s3_bucket" "bucket" {
  bucket = var.bucket_name
  acl    = "private"

  lifecycle {
    prevent_destroy = false
  }

  tags = merge(var.tags, { Name = "s3-bucket" })
}

output "bucket_id" {
  value = aws_s3_bucket.bucket.id
}

#####################################################
# modules/iam/main.tf
#####################################################
resource "aws_iam_user" "user" {
  name = var.user_name
  tags = var.tags
}

resource "aws_iam_policy" "policy" {
  name        = var.policy_name
  description = "IAM policy for user"
  policy      = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Action   = ["s3:*"]
        Effect   = "Allow"
        Resource = "*"
      }
    ]
  })
```

```
}

resource "aws_iam_user_policy_attachment" "attach" {
  user       = aws_iam_user.user.name
  policy_arn = aws_iam_policy.policy.arn
}

output "user_arn" {
  value = aws_iam_user.user.arn
}

#####################################################
# dev.tfvars
#####################################################
region            = "ap-south-1"
vpc_cidr          = "10.0.0.0/16"
public_subnet_cidr= "10.0.1.0/24"
instance_type     = "t2.micro"
key_name          = "dev-key"
ami_id            = "ami-0abcdef1234567890"
s3_bucket_name    = "dev-bucket-terraform"
iam_user_name     = "dev-user"
iam_policy_name   = "dev-policy"

#####################################################
# test.tfvars
#####################################################
region            = "ap-south-1"
vpc_cidr          = "10.1.0.0/16"
public_subnet_cidr= "10.1.1.0/24"
instance_type     = "t2.small"
key_name          = "test-key"
ami_id            = "ami-0abcdef1234567890"
s3_bucket_name    = "test-bucket-terraform"
iam_user_name     = "test-user"
iam_policy_name   = "test-policy"
```