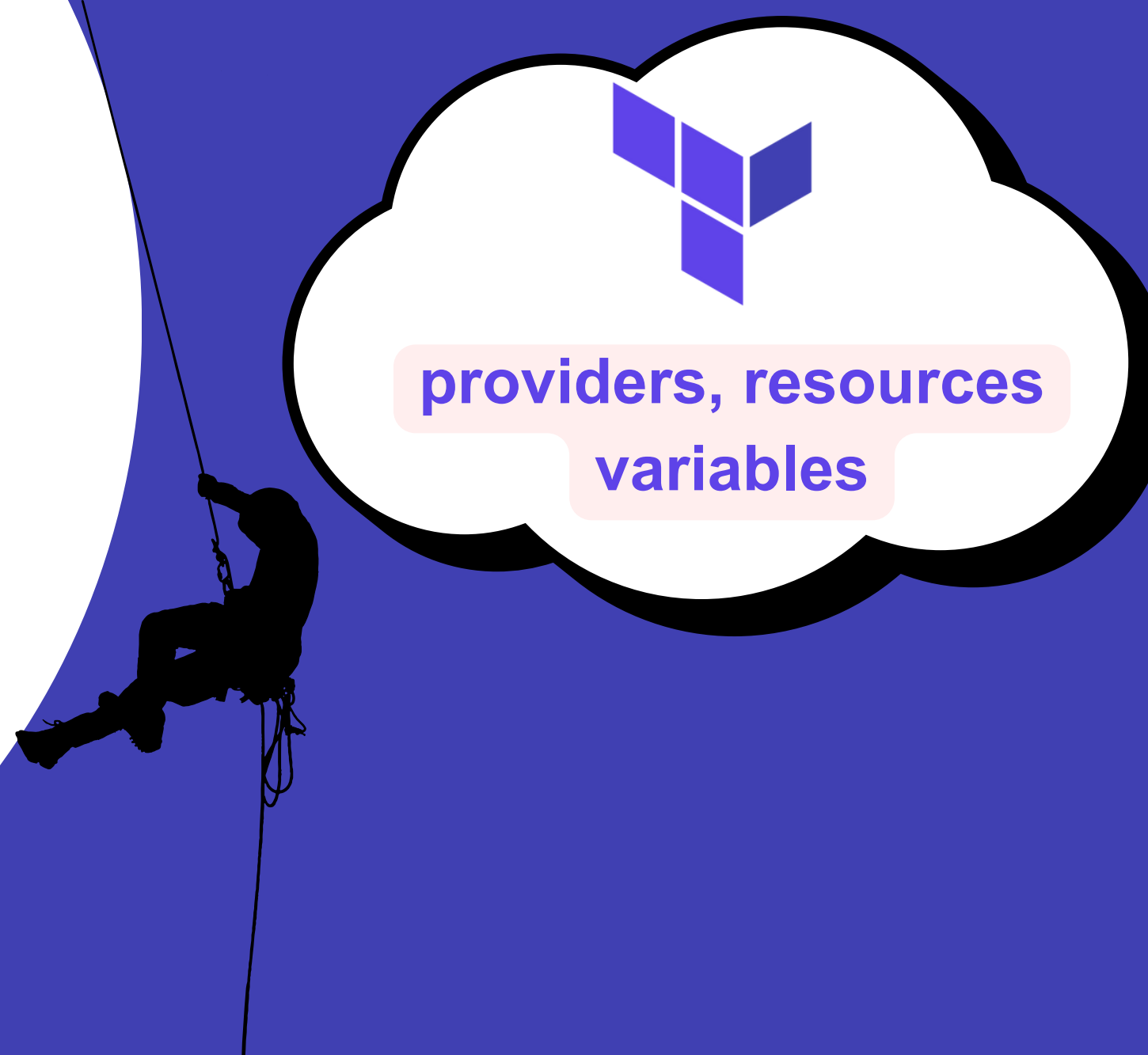
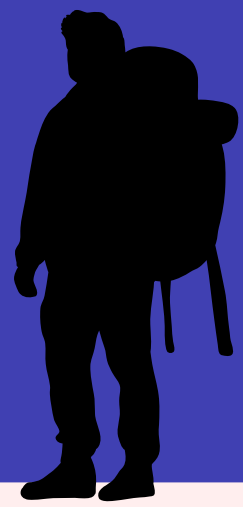
terraform file compo nents



Ömer Berat Sezer

PhD | Software Engineer | DevOps | Solution Architect

linkedin.com/in/omerberatsezer/



github.com/omerbsezer

Terraform file has different components to define infrastructure for different purposes

After creating files, Terraform commands:

- init: downloads the required executable apps dependent on providers.
- plan: dry-run for the infrastructure, not actually running/provisioning the infra
- apply: runs/provisions the infrastructure
- destroy: deletes the infrastructure

Providers

Terraform downloads required executable files from own cloud to run Infrastructure as Code (IaC) for the corresponding providers

linkedin.com/in/omerberatsezer

Providers

for AWS

providers.tf

```
terraform {
  required_providers  {
     aws = {
        source   =  "hashicorp/aws"    # <=  AWS library
        version  =  "~> 5.82.2"        # <= AWS version
     }
  }
}
provider “aws” {
     region  =  "us-east-1"
}
```

linkedin.com/in/omerberatsezer

Providers

for Azure

providers.tf

```
terraform {
  required_providers  {
     azurerm = {
        source   =  "hashicorp/azurerm"     # <= Azure library
        version  =  "= 4.14.0"              # <= version
     }
  }
}
provider “azurerm” {
      features { }
}
```

linkedin.com/in/omerberatsezer

Providers

Google Cloud (GCP)

providers.tf

```
terraform {
  required_providers  {
    google = {
      source   =  "hashicorp/google"    # <= GCP library
      version  =  "6.14.1"              # <= version
    }
  }
}
provider “google” {
    project  =  "my-project-id"
    region   =  "us-central1"
}
```

linkedin.com/in/omerberatsezer

Resources

Resources are used to define for different cloud components and objects like EC2 instances, VPC, Route Table, Subnets,
Lambda, API Gateway,
S3 Buckets, etc.

Resources are defined according to cloud provider design and attributes.

linkedin.com/in/omerberatsezer

Resources

for AWS

resources.tf

```
resource  "aws_vpc" "my_vpc" {        # <= resource | type | variable
     cidr_block  =  "172.16.0.0/16"    # <= awc_vpc attributes; cidr
     tags = {                          # <=  attributes; tag
        Name  =  "tf-example"
     }
}
 resource "aws_subnet" "my_subnet" {
      vpc_id                  =  aws_vpc.my_vpc.id    # <= values
      cidr_block              =  "172.16.0.0/16"
      availability_zone   =  "us-east-2a"
}
```

https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance

linkedin.com/in/omerberatsezer

Resources

for Azure

resources.tf

```
resource "azurerm_resource_group" "example" {
    name     = "example-resources"    # <= attributes are different
    location = "West Europe"          #    for each Cloud, and resource
}
resource "azurerm_virtual_network" "example" {
    name                = "example-network"
    resource_group_name =  azurerm_resource_group.example.name
    location            = azurerm_resource_group.example.location
    address_space       = ["10.0.0.0/16"]
}
```

https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs

linkedin.com/in/omerberatsezer

Resources

for Google Cloud

resources.tf

```
resource "google_compute_subnetwork" "network-with-private-secondary-ip-ranges" {
    name          = "test-subnetwork"
    ip_cidr_range = "10.2.0.0/16"
    region        = "us-central1"
    network       = google_compute_network.custom-test.id
    secondary_ip_range {
        range_name    = "tf-test-secondary-range-update1"
        ip_cidr_range = "192.168.10.0/24"
    }
}
```

https://registry.terraform.io/providers/hashicorp/google/latest/docs

linkedin.com/in/omerberatsezer

Variables (string)

Variables help to avoid hard coding on the infrastructure code

variables.tfvars

```
variable "instance_type" {
    type          =  string                    # <= type string
    description   =  "EC2 Instance Type"
}
output "instance_type_output" {
     value        =  var.instance_type
}
```

linkedin.com/in/omerberatsezer

Variables (number)

variables.tfvars

```
variable  "instance_count" {
    type          =  number                    # <= type number
    default       =  3
}
output "instance_count_output" {
    value         =  var.instance_count
}
```

linkedin.com/in/omerberatsezer

Variables (bool)

variables.tfvars

```
variable "enable_instance" {
    type          = bool
    default       = true
}
output "enable_instance_output" {
    value         =  var.enable_instance
}
```

linkedin.com/in/omerberatsezer

Variables
(list)

variables.tfvars

```
variable  "availability_zones" {
    type            =  list(string)
    default         =  ["us-east-1a", "us-east-1b", "us-east-1c" ]
}
output "availability_zones_output" {
     value          =   var.availability_zones
}
```

linkedin.com/in/omerberatsezer

Variables (tuple)

variables.tfvars

```
variable "instance_config" {
    type          = tuple([string, string])
    default       = ["t2.micro", "ami-0c55b159cbfafe1f0"]
}
output "instance_config_output" {
     value          =  var.instance_config
}
```

linkedin.com/in/omerberatsezer

Variables (map)

variables.tfvars

```
variable "instance_config" {
    type          = map(string)
    default       = {
        instance_type  =  "t2.micro"
        ami_id         =  "ami-0c55b159cbfafe1f0"
    }
}
resource "aws_instance" "example" {
    ami           = var.instance_config["ami_id"]
    instance_type = var.instance_config["instance_type"]
}
```

linkedin.com/in/omerberatsezer

Variables

In real use-case scenario, variables are stored in one file, then while running command, variables.tf is used as input file

```
user@terraform:$ terraform plan --var-file="variables.tfvars"
user@terraform:$ terraform apply --var-file="variables.tfvars"
```

linkedin.com/in/omerberatsezer

```
variable "instance_type" {
    type = string
    description = "EC2 Instance Type"
}

variable "tag" {
    type = string
    description = "The tag for the EC2 instance"
}

variable "location" {
    type = string
    description = "The project region"
    default = "eu-central-1"
}

variable  "availability_zone" {
    type = string
    description = "The project availability zone"
    default = "eu-central-1c"
}

variable "ami" {
    type = string
    description = "The project region"
}
```

```
instance_type     =   "t2.nano"
tag               =   "EC2 Instance for DEV"
location          =   "eu-central-1"
availability_zone =   "eu-central-1c"
ami               =   "ami-0e067cc8a2b58de59" # Ubuntu 20.04 eu-central-1 Frankfurt
```

linkedin.com/in/omerberatsezer

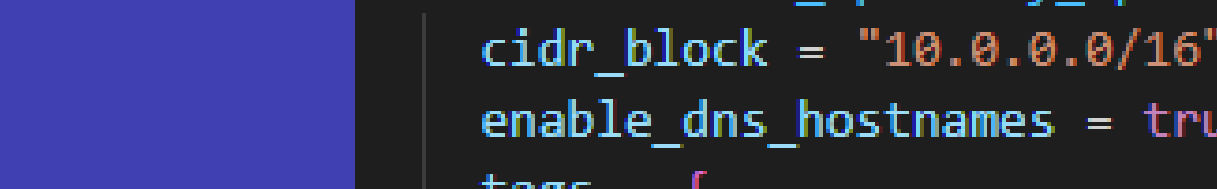
```hcl
provider "aws" {
  region     = var.location
}

locals {
  staging_env = "staging"
}

resource "aws_vpc" "my_vpc" {
  cidr_block = "10.0.0.0/16"
  enable_dns_hostnames = true
  tags = {
    Name = "${local.staging_env}-vpc-tag"
  }
}

resource "aws_subnet" "my_subnet" {
  vpc_id = aws_vpc.my_vpc.id
  cidr_block = "10.0.0.0/16"
  availability_zone = var.availability_zone
  tags = {
    Name = "${local.staging_env}-subnet-tag"
  }
}
```

variables from file
with
var => variable
location in variable file



linkedin.com/in/omerberatsezer

user@terraform:$ ##############################

Follow for Tips on AWS, K8s, Docker, Linux
Terraform, Ansible, DevOps, AI/ML
Why? Cause; More will unfold over time
||
V
https://linkedin.com/in/omerberatsezer

https://github.com/omerbsezer

Feel free to like, share, or repost
to help more people see it

user@terraform:$ ##############################

linkedin.com/in/omerberatsezer