Module 3

Terraform Providers

HashiCorp
Terraform

Agenda

- Introduction to Providers
- Local Provider
- Public Cloud Providers
- AWS resource examples
- Azure resource examples
- Google Cloud resource examples
- Lab 3

Introduction to Providers

- A provider in Terraform is a plugin that enables interaction with an API. The providers are specified in the Terraform configuration code. They tell Terraform which services it needs to interact with.
- Terraform currently has 1700+ providers
- Each provider adds a set of resource types and/or data sources that Terraform can manage.

Local Provider

- The Local provider is used to manage local resources, such as files.
- Terraform primarily deals with remote resources which are able to outlive a single Terraform run, and so local resources can sometimes violate its assumptions.
- Local resources are best used with care, since depending on local state can make it hard to apply the same Terraform configuration on many different local systems where the local resources may not be universally available

```
resource "local_file" "foo" {
  content  = "foo!"
  filename = "${path.module}/foo.bar"
}
```

```
data "local_file" "foo" {
  filename = "${path.module}/foo.bar"
}

resource "aws_s3_object" "shared_zip" {
  bucket  = "my-bucket"
  key     = "my-key"
  content = data.local_file.foo.content
}
```

HashiCorp Terraform

Public Cloud Providers

- Terraform can provision infrastructure across public cloud providers such as Amazon Web Services (AWS), Azure and Google Cloud, as well as private cloud and virtualization platforms such as OpenStack and VMWare.
- Providers differ! Always refer to documentation regarding terminology and resource types and features

What cloud providers does Terraform support?

Providers

- AWS.
- Azure.
- Google Cloud Platform.
- Kubernetes.
- Alibaba Cloud.
- Oracle Cloud Infrastructure.

HashiCorp Terraform

Aliases

- You can optionally define multiple configurations for the same provider, and select which one to use on a per-resource or per-module basis. The primary reason for this is to support multiple regions for a cloud platform; other examples include targeting multiple Docker hosts, multiple Consul hosts, etc.

```
# The default provider configuration; resources that begin with `aws_` will u
# it as the default, and it can be referenced as `aws`.
provider "aws" {
  region = "us-east-1"
}

# Additional provider configuration for west coast region; resources can
# reference this as `aws.west`.
provider "aws" {
  alias  = "west"
  region = "us-west-2"
}
```

HashiCorp Terraform

AWS resource example

Here is a sample deployment of a VPC with subnets, routing tables and routing table associations.

Note: *The Internet Gateway, NAT Gateway and Security group resources have been omitted for brevity*

```
resource "aws_vpc" “prod" {
   cidr_block = "10.1.0.0/16"
}

resource "aws_subnet" “public" {
   vpc_id     = aws_vpc.prod.id
   cidr_block = "10.1.1.0/24"

   tags = {
   Name = “Public"
  }
}

resource "aws_subnet" “private" {
   vpc_id     = aws_vpc.prod.id
   cidr_block = "10.1.2.0/24"

   tags = {
   Name = “Private"
  }
}

resource "aws_route_table_association" "a" {
   subnet_id      = aws_subnet.public.id
   route_table_id =
aws_route_table.public_rt.id
}

resource "aws_route_table_association" “b" {
   subnet_id      = aws_subnet.private.id
   route_table_id =
aws_route_table.private_rt.id
}

resource "aws_route_table" “public_rt" {
   vpc_id = aws_vpc.prod.id

   route {
      cidr_block = “0.0.0.0/0"

gateway_id=aws_internet_gateway.myigw.id
  }

resource "aws_route_table" “private_rt" {
   vpc_id = aws_vpc.prod.id

   route {
      cidr_block = “0.0.0.0/0"

gateway_id=aws_nat_gateway.mynatgtw.id
  }
```

HashiCorp Terraform

```
resource "aws_vpc" “prod" {
   cidr_block = "10.1.0.0/16"
}

resource "aws_subnet" “public" {
   vpc_id     = aws_vpc.prod.id
   cidr_block = "10.1.1.0/24"

   tags = {
   Name = “Public"
  }
}

resource "aws_subnet" “private" {
   vpc_id     = aws_vpc.prod.id
   cidr_block = "10.1.2.0/24"

   tags = {
   Name = “Private"
  }
}

resource "aws_route_table_association" "a" {
   subnet_id      = aws_subnet.public.id
   route_table_id =
aws_route_table.public_rt.id
}

resource "aws_route_table_association" “b" {
   subnet_id      = aws_subnet.private.id
   route_table_id =
aws_route_table.private_rt.id
}

resource "aws_route_table" “public_rt" {
   vpc_id = aws_vpc.prod.id

   route {
      cidr_block = “0.0.0.0/0"

gateway_id=aws_internet_gateway.myigw.id
  }

resource "aws_route_table" “private_rt" {
   vpc_id = aws_vpc.prod.id

   route {
      cidr_block = “0.0.0.0/0"

gateway_id=aws_nat_gateway.mynatgtw.id
  }
```

```
resource "aws_vpc" “prod" {
   cidr_block = "10.1.0.0/16"
}
```

```
resource "aws_subnet" “public" {
   vpc_id     = aws_vpc.prod.id
   cidr_block = "10.1.1.0/24"

   tags = {
   Name = “Public"
  }
}
```

```
resource "aws_subnet" “private" {
   vpc_id     = aws_vpc.prod.id
   cidr_block = "10.1.2.0/24"

   tags = {
   Name = “Private"
  }
}
```

```
resource "aws_route_table_association" "a" {
   subnet_id      = aws_subnet.public.id
   route_table_id =
aws_route_table.public_rt.id
}
```

```
resource "aws_route_table_association" “b" {
   subnet_id      = aws_subnet.private.id
   route_table_id =
aws_route_table.private_rt.id
}
```

```
resource "aws_route_table" “public_rt" {
   vpc_id = aws_vpc.prod.id

   route {
      cidr_block = “0.0.0.0/0"

gateway_id=aws_internet_gateway.myigw.id
  }
```

```
resource "aws_route_table" “private_rt" {
   vpc_id = aws_vpc.prod.id

   route {
      cidr_block = “0.0.0.0/0"

gateway_id=aws_nat_gateway.mynatgtw.id
  }
```

HashiCorp
Terraform

```
resource "aws_vpc" “prod" {
   cidr_block = "10.1.0.0/16"
}
```

```
resource "aws_subnet" “public" {
   vpc_id     = aws_vpc.prod.id
   cidr_block = "10.1.1.0/24"

   tags = {
   Name = “Public"
  }
}
```

```
resource "aws_route_table_association" "a" {
   subnet_id      = aws_subnet.public.id
   route_table_id = aws_route_table.public_rt.id
}
```

```
resource "aws_route_table" “public_rt" {
   vpc_id = aws_vpc.prod.id

   route {
      cidr_block = “0.0.0.0/0"
      gateway_id=aws_internet_gateway.myigw.id
  }
```

```
resource "aws_subnet" “private" {
   vpc_id     = aws_vpc.prod.id
   cidr_block = "10.1.2.0/24"

   tags = {
   Name = “Private"
  }
}
```

```
resource "aws_route_table_association" "a" {
   subnet_id      = aws_subnet.private.id
   route_table_id = aws_route_table.private_rt.id
}
```

```
resource "aws_route_table" “private_rt" {
   vpc_id = aws_vpc.prod.id

   route {
      cidr_block = “0.0.0.0/0"
      gateway_id=aws_nat_gateway.mynatgtw.id
  }
```

```
resource "aws_vpc" "prod" {
  cidr_block = "10.1.0.0/16"
}
```

```
resource "aws_subnet" "public" {
  vpc_id     = aws_vpc.prod.id
  cidr_block = "10.1.1.0/24"

  tags = {
  Name = "Public"
  }
}
```

```
resource "aws_route_table_association" "a" {
  subnet_id      = aws_subnet.public.id
  route_table_id = aws_route_table.public_rt.id
}
```

```
resource "aws_route_table" "public_rt" {
  vpc_id = aws_vpc.prod.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id=aws_internet_gateway.myigw.id
  }
```

```
resource "aws_subnet" "private" {
  vpc_id     = aws_vpc.prod.id
  cidr_block = "10.1.2.0/24"

  tags = {
  Name = "Private"
  }
}
```

```
resource "aws_route_table_association" "a" {
  subnet_id      = aws_subnet.private.id
  route_table_id = aws_route_table.private_rt.id
}
```

```
resource "aws_route_table" "private_rt" {
  vpc_id = aws_vpc.prod.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id=aws_nat_gateway.mynatgtw.id
  }
```

HashiCorp Terraform

```
resource "aws_vpc" “prod" {
   cidr_block = "10.1.0.0/16"
}
```

```
resource "aws_subnet" “public" {
   vpc_id     = aws_vpc.prod.id
   cidr_block = "10.1.1.0/24"

   tags = {
   Name = “Public"
  }
}
```

```
resource "aws_route_table_association" "a" {
   subnet_id      = aws_subnet.public.id
   route_table_id = aws_route_table.public_rt.id
}
```

```
resource "aws_route_table" “public_rt" {
   vpc_id = aws_vpc.prod.id

   route {
      cidr_block = “0.0.0.0/0"
      gateway_id=aws_internet_gateway.myigw.id
   }
```

```
resource "aws_subnet" “private" {
   vpc_id     = aws_vpc.prod.id
   cidr_block = "10.1.2.0/24"

   tags = {
   Name = “Private"
  }
}
```

```
resource "aws_route_table_association" "a" {
   subnet_id      = aws_subnet.private.id
   route_table_id = aws_route_table.private_rt.id
}
```

```
resource "aws_route_table" “private_rt" {
   vpc_id = aws_vpc.prod.id

   route {
      cidr_block = “0.0.0.0/0"
      gateway_id=aws_nat_gateway.mynatgtw.id
   }
```

Terraform registry: aws

Providers / hashicorp / aws / Version 4.56.0 Latest Version

aws

Overview Documentation USE PROVIDER

AWS DOCUMENTATION

Filter

- aws provider
- Guides
- ACM (Certificate Manager)
- ACM PCA (Certificate Manager Private Certificate Authority)
- AMP (Managed Prometheus)
- API Gateway
- API Gateway V2
- Account Management
- Amplify
- App Mesh

AWS Provider

Use the Amazon Web Services (AWS) provider to interact with the many resources supported by AWS. You must configure the provider with the pr credentials before you can use it.

Use the navigation to the left to read about the available resources.

To learn the basics of Terraform using this provider, follow the hands-on g started tutorials. Interact with AWS services, including Lambda, RDS, and I following the AWS services tutorials.

Example Usage

Terraform 0.13 and later:

How to use this provider

To install this provider, copy and paste this code into your Terraform configuration. Then, run `terraform init`.

Terraform 0.13+

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "4.56.0"
    }
  }
}

provider "aws" {
  # Configuration options
}
```

HashiCorp Terraform

Azure resource example

An Azure
Virtual Network

```
resource "azurerm_virtual_network" “my_vnet" {
   name              = “pro-network"
   location          =   azurerm_resource_group.rg1.location
   resource_group_name = azurerm_resource_group.rg1.name
   address_space       = ["10.1.0.0/16"]
   dns_servers         = ["10.1.0.4", "10.1.0.5"]

   subnet {
       name          = "subnet1"
       address_prefix = "10.1.1.0/24"
   }

   subnet {
       name          = "subnet2"
       address_prefix = "10.1.2.0/24"
       security_group = azurerm_network_security_group.nsg1.id
   }
```

```
resource "azurerm_resource_group" “rg1" {
   name     = “prod-resources"
   location = "West Europe"
}
```

```
resource "azurerm_network_security_group" “nsg1" {
   name              = “web-security-group"
   location          = azurerm_resource_group.rg1.location
   resource_group_name = azurerm_resource_group.rg1.name
}
```

Terraform registry: azurerm

Providers / hashicorp / azurerm / Version 3.45.0 Latest Version

azurerm

Overview Documentation USE PROVIDER

AZURERM DOCUMENTATION

Filter

- azurerm provider
- Guides
- AAD B2C
- API Management
- Active Directory Domain Services
- Advisor
- Analysis Services
- App Configuration
- App Service (Web Apps)
- Application Insights
- Attestation

Azure Provider

The Azure Provider can be used to configure infrastructure in Microsoft A using the Azure Resource Manager API's. Documentation regarding the D Sources and Resources supported by the Azure Provider can be found in t navigation to the left.

To learn the basics of Terraform using this provider, follow the hands-on g started tutorials.

Interested in the provider's latest features, or want to make sure you're up date? Check out the changelog for version information and release notes.

Authenticating to Azure

How to use this provider

To install this provider, copy and paste this code into your Terraform configuration. Then, run `terraform init`.

Terraform 0.13+

```
terraform {
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = "3.45.0"
    }
  }
}

provider "azurerm" {
  # Configuration options
}
```

HashiCorp Terraform

Google Cloud resource example

A Google Cloud VPC

```
resource "google_compute_network" "custom-test" {
  name                    = "test-network"
  auto_create_subnetworks = false
}
```

```
resource "google_compute_subnetwork" “subnet-demo" {
   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"
   }
}
```

Terraform registry: google

Providers / hashicorp / google / Version 4.54.0 Latest Version

google

Overview Documentation USE PROVIDER

google
by: HashiCorp
Official

243.6M Installs

hashicorp/terraform-provider-google

LATEST VERSION
4.54.0 Published 5 days ago

- API Gateway
- Access Approval
- Access Context Manager (VPC Service Controls)
- AlloyDB
- Apigee
- Apikeys
- App Engine
- Artifact Registry

Google Cloud Platform Provider

The Google provider is used to configure your Google Cloud Platform infrastructure. See the Getting Started page for an introduction to using t provider.

To learn the basics of Terraform using this provider, follow the hands-on g started tutorials. For more involved examples, try provisioning a GKE clust deploying Consul-backed Vault into it using Terraform Cloud.

A typical provider configuration will look something like:

```
provider "google" {
  project     = "my-project-id"
  region      = "us-central1"
}
```

How to use this provider

To install this provider, copy and paste this code into your Terraform configuration. Then, run `terraform init`.

Terraform 0.13+

```
terraform {
  required_providers {
    google = {
      source = "hashicorp/google"
      version = "4.54.0"
    }
  }
}

provider "google" {
  # Configuration options
}
```

HashiCorp Terraform

Lab Group Discussion & Explore

Scoping

- What Type of Resources are you going to create ?
- What specific Properties do we need to consider for these resources ?

Workflow

- Are there any Implicit or Explicit dependencies ?
- What TF *Provider*, *Resource,* and *Modules* are required ?

Research

- Identify the TF documentation that will assist in developing the solution

Lab 3

Create a basic VPC deployment on AWS

In this lab you will deploy a simple VPC architecture comprising of a VPC, subnets, routing tables, gateways and security groups.

HashiCorp Terraform

Any questions…