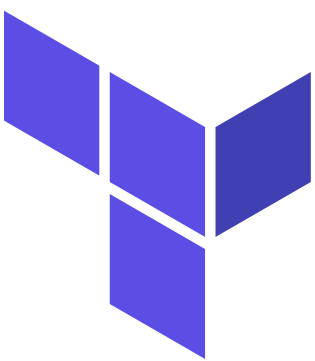
Terraform Basics

Be smart. Think open source.



Write, Plan, and Create Infrastructure as Code

HashiCorp

Terraform

Agenda

- Introduction to Terraform
- Setup
- HCL
- My first Terraform
- Configuration Objects

Introduction to Terraform

What is it?

Facts

- First release: 2014-07-28
- Written by: HashiCorp
- Written in: Go

Wikipedia says...

It allows users to define a datacenter infrastructure in a high-level configuration language, from which it can create an execution plan to build the infrastructure[...].

it essentially is

Infrastructure as Code

Setup

https://learn.hashicorp.com/terraform/getting-started/install.html

Check

```
> terraform

Usage: terraform [-version] [-help] <command> [args]

The available commands for execution are listed below.
The most common, useful commands are shown first, followed by
less common or more advanced commands. If you're just getting
started with Terraform, stick with the common commands. For the
other commands, please read the help and docs before usage.

Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Workspace management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a Terraform working directory
    output             Read an output from a state file
    plan               Generate and show an execution plan
    providers          Prints a tree of the providers used in the configuration
    push               Upload this Terraform module to Atlas to run
    refresh            Update local state file against real resources
    show               Inspect Terraform state or plan
    taint              Manually mark a resource for recreation
    untaint            Manually unmark a resource as tainted
    validate           Validates the Terraform files
    version            Prints the Terraform version
    workspace          Workspace management

All other commands:
    debug              Debug output management (experimental)
    force-unlock       Manually unlock the terraform state
    state              Advanced state management
```

HCL

HashiCorp configuration language https://github.com/hashicorp/hcl

HCL/JSON

```
resource "azurerm_resource_group" "workshoptest-01" {
  name     = "rg-adsy-workshoptest-01"
  location = "westeurope"
}
```

```
{
  "resource": {
    "azurerm_resource_group": {
      "workshoptest-01": {
        "name": "rg-adsy-workshoptest-01",
        "location": "westeurope"
      }

    }
  }
}
```

My first Terraform

main.tf

```
resource "azurerm_resource_group" "main" {
  name     = "main"
  location = "${var.location}"
}

resource "azurerm_virtual_network" "main" {
  name                = "main_network"
  address_space       = ["10.0.0.0/16"]
  location            = "${azurerm_resource_group.main.location}"
  resource_group_name = "${azurerm_resource_group.main.name}"
}
```

variables.tf

```
variable "location" {
  default   = "West Europe"
}
```

output.tf

```
output "rgname" {
  description       = "The name of our resource group"
  value             = "${azurerm_resource_group.main.name}"
}
```

Try it

```
> terraform init
> terraform plan
```

Adapt it

https://www.terraform.io/docs/providers/index.html

Configuration Objects

- resource
- data
- provider
- variable
- output
- locals
- module
- terraform

resource

```
resource "azurerm_resource_group" "main" {
  name        = "my-resource-group"
  location    = "West US 2"
}
```

Defines an infrastructure resource

https://www.terraform.io/docs/configuration/resources.html

data

```
data "azurerm_public_ip" "mypubip" {
  name        = "${azurerm_public_ip.ip-01.name}"
  resource_group_name   = "${azurerm_virtual_machine.myhost-01.resource_group_name}"
}
```

Defines a data source that can be reused. Must be unique in combination of <TYPE> and <NAME>.

https://www.terraform.io/docs/configuration/data-sources.html

provider

```
provider "azurerm" {
  version     = "=1.23.0"
}
```

Defines providers to use. Version pinning is recommended.

https://www.terraform.io/docs/configuration/providers.html

variable

```
variable "user_name" {
  type        = "string"
  default     = "user"
  description = "The Username of our new user"
}
```

Defines variables. Can be of different types (string, map, list or boolean)

https://www.terraform.io/docs/configuration/variables.html

output

```
output "address" {
  value       = "${data.azurerm_public_ip.mypubip.ip_address}"
  description = "The IP address of our new host"
}
```

Defines data outputs. Used for automation and collecting information.

https://www.terraform.io/docs/configuration/outputs.html

locals

```
locals {
  user_name   = "user"
}
```

Defines local variables inside a module.

https://www.terraform.io/docs/configuration/locals.html

module

```
module "akscluster" {
 source       = "azure/aks/defaultcluster"
 version      = "1.1.0"
 nodes        = 6
}
```

Defines a terraform module. Variables can be passed to the module. Version pinning is recommended.

https://www.terraform.io/docs/configuration/modules.html

terraform

```
terraform {
  required_version = "> 0.7.0"
}
```

Defines terraform configuration.

https://www.terraform.io/docs/configuration/terraform.html

meta-parameters

Can be applied to all resource definitions regardless of type

- count (not applicable to modules)
- depends_on (not applicable to modules)
- provider
- lifecycle
- create_before_destroy
- prevent_destroy
- ignore_changes

count (not applicable to modules)

```
resource "azurerm_virtual_machine" "main" {
  count = 3 # creates 3 virtual machines
  name = "VM-${count.index}"
}
```

Creates multiple instances of the resource

depends_on (not applicable to modules)

```
resource "azurerm_virtual_machine" "main" {
  # forces creation of module components before this VM
  depends_on = [ "${module.aks.fqdn}" ]
}
```

Creates a dependency when default dependency management fails

provider

```
provider "azurerm" {
  alias = "us"
  location = "westus"
  version = "~> 1.23.0"
}

resource "azurerm_virtual_machine" "us" {
  # force usage of the westus-provider
  provider = "azurerm.us"
}
```

Specifies the provider to use. Makes most sense when the same provider is used multiple times (https://www.terraform.io/docs/configuration/resources.html#multiple-provider-instances)

lifecycle

Lifecycle behaviour of the resource. Knows 3 attributes:

- create_before_destroy
- prevent_changes
- ignore_changes

create_before_destroy

```
resource "azurerm_dns_a_record" "website" {
  create_before_destroy = true
}
```

Force creation of a new resource *before* the old resource is deleted. Useful for example for DNS records.

prevent_destroy

```
resource "azurerm_kubernetes_cluster" "main" {
  prevent_destroy = true
}
```

Any plan that wants to destroy this resource will fail.

ignore_changes

```
resource "azurerm_kubernetes_cluster" "main" {
  ignore_changes = [ "vm_size" ]
}
```

When one of the specified attributes change no action will be taken.

Providers

A provider connects terraform configuration with a corresponding API

Different Providers

- Azure
- AWS
- Google Cloud
- Github
- Kubernetes
- PowerDNS
- MySQL
- much more... https://www.terraform.io/docs/providers/index.html

First steps with Github

main.tf

```
provider "github" {
  token        = "${var.github_token}"
  organization  = "${var.github_organization}"
  version       = "~> 1.3"
}


resource "github_repository" "tf-example" {
  name          = "${var.github_repo_name}"
  description   = "${var.github_repo_desc}"
}
```

variables.tf

```
variable "github_token" {
  default = "abcd"
}

variable "github_organization" {
  default = "myorga"
}

variable "github_repo_name" {
  default "tf-test"
}

variable "github_repo_desc" {
  default "test-repo for terraform"
}
```

myrepo.tfvars

```
github_token = "w841ce33fcfc1wde71fdb50c4dd852c63108b118"
github_orga = "adfinis-sygroup"
github_repo_name = "tf-test"
github_repo_desc = "testdescription"
```

Try it!

```
terraform init
terraform plan -var-file=myrepo.tfvars
terraform apply -var-file=myrepo.tfvars
```

More information

https://www.terraform.io/docs

Attribution / License

Feel Free to Contact Us

www.adfinis-sygroup.ch

Tech Blog

GitHub

info@adfinis-sygroup.ch

Twitter