Terraform part

Mateusz Lewicki

Terraform part

Mateusz Lewicki

sierpnia 11, 2020

Puprouse of using Terraform

Terraform by Hashicorp is infrastructure description langauage in HLC format (Hashicorp DSL) used in widely name CI/CD for construction of all needed infrastructure components for application by using principles of infrastructure as code.

Brief of terrafom in current project

Project consist of 7/8 terraform files (7 as code and 8th for state):

- variables.tf
- images.tf
- terraform.tfvars
- terraform.tfstate
- outputs.tf
- networks.tf
- main.tf

In this project Terraform is used to build “2 tier” infrastructure with loadbalancing in docker.

Terrraform code was split into parts (see above).
Terraform by default is looking through directory for every Terraform related files.
Terrafom does not need to have entrypoint file, but it is good practice to have main file.
Without backend declaration Terraform is using by default `local` provider which means that state file will be created in the same directory

File description

main.tf (part of)

```
provider "docker" {
}


resource "docker_container" "LB" {
  image = docker_image.lb.latest
  name  = "LB"
  networks_advanced{
      name=docker_network.public_network.name
  }
  networks_advanced{
      name=docker_network.app_network_1.name
  }
  networks_advanced{
      name=docker_network.app_network_2.name
  }
  ports{
      internal= var.http_port
      external= var.http_port
  }
  ports{
      internal = var.https_port
      external = var.https_port
  }
   ports{
      internal = 1936
      external = 1936
  }
  host{
    host="app1"
    ip=docker_container.apache_1.ip_address
```

```
  }
  host{
    host="app2"
    ip=docker_container.apache_2.ip_address
  }
}

[...]
```

This file consist of provider and resource declaration blocks used by Terraform Configuration Language and Docker Provider

Overwiew

main.tf file is (as the name suggests) the main file of whole infrastructure builded in this project. At the top we have `provider` block it is saying that we want to use a docker provider in the latest version (as we don't specified the version)
Here we have a snippet of on of the `docker_container` resource with identifier `LB`. This resource is part of Docker provider described above. It's built with several fields and block:

`image = docker_image.lb.latest` - (type: String)

It is a name (or project/name) of the docker image to use. In this case we are referring to `docker_image` resource with `lb` identifier and the `latest` version. `.latest` can be replaced by `:latest` tag

`name = "LB"` - (type: String)

Literal name

`networks_advanced{name=docker_network.public_network.name }` - (type:block->String)

Block where we define to which network container should be attached (one block for one network).
Inside we have `name` field where we ae using name of network (in this case public) from resource `docker_network` defined in *networks.tf* file

`ports{ internal= var.http_port external= var.http_port }` - (type:block->Int,Int)

Block with definition which ports should be exposed (one block for one ports pair). Here we are using ports from variable `http_port`

`host{ host="app1" ip=docker_container.apache_1.ip_address }-` (type:block->String,String)

Block used for additional `/etc/hosts` entries. Here we are using `ip_address` from `apache_1` the `docker_container` resource

images.tf

```
resource "docker_image" "alpine" {
  name = "alpine:latest"
  keep_locally = true
}

resource "docker_image" "web" {
  name = "${var.registry}/${var.web_image_name}"
  keep_locally = true
}

resource "docker_image" "db" {
  name = "${var.registry}/${var.db_image_name}"
  keep_locally = true
}

resource "docker_image" "lb" {
  name = "${var.registry}/${var.lb_image_name}"
  keep_locally = true
}
```

This file consist of resource declaration blocks used by Docker Provider

Overwiew

images.tf file contains definitions of container images, which we can then use in `docker_container` resource.
File defines several `docker_image` blocks one for single image. This block in this case has two fields:

`name = "${var.registry}/${var.web_image_name}"` - (type: String)

This field defines which image to use. In this particural entry we use custom registry and image name defined in own variables.
In the first resource block in source file you can find literall name of image `alpine:latest`

`keep_locally = true` - (type: Boolean)

This field is informing provider to not remove images when `destroy` command is provided.

networks.tf

```
resource "docker_network" "app_network_1" {
  name = "app_network_1"
  internal = true
  ipam_config{
      subnet="10.0.0.0/28"
  }
}

resource "docker_network" "app_network_2" {
  name = "app_network_2"
  internal = true
  ipam_config{
      subnet="10.0.0.16/28"
  }
}
resource "docker_network" "public_network" {
  name = "public_network"
  ipam_config{
      subnet="10.0.0.32/28"
  }
}
```

This file consist of resource declaration blocks used by Docker Provider

variables.tf

```
variable "http_port" {
  type        = string
}

variable "https_port" {
  type        = string
}

variable "db_port" {
  type        = string
}

variable "registry" {
  type        = string
```

```
}

variable "db_image_name" {
  type        = string
}
variable "web_image_name" {
  type        = string
}
variable "lb_image_name" {
  type        = string
}
```

This file consist of variable declaration blocks used by Terraform Configuration Language

outputs.tf

```
output "apache_1_ip_addr" {
  value = docker_container.apache_1.ip_address
}

output "apache_2_ip_addr" {
  value = docker_container.apache_2.ip_address
}
output "LB_ip_addr" {
  value = docker_container.LB.ip_address
}
```

This file consist of output declaration blocks used by Terraform Configuration Language

terraform.tfvars

```
  http_port = "80"
  https_port = "443"
  db_port = "3306"
  registry = "localhost:5000"
  web_image_name = "lamp_terr/web"
  db_image_name = "lamp_terr/database"
  lb_image_name = "lamp_terr/loadbalancer"
```

This file consist of variable definitions