Adding Maps and Lookups in your Terraform files

The lab files can be found @ https://github.com/satyensingh/terraform-assignment-resources.git

In the repo you will find main.tf, outputs.tf, and variables.tf.

Add a new variable called env. Set a description to “env: dev or prod”.

variable "env" {

description = "env:dev or prod"

}

--------------------------------------------------------------------------------------------------

Convert the type from image\_name to map.

Change the default to use key/value pairs. Set dev to ghost:latest and prod to ghost:alpine.

variable "image\_name" {

type = map

description = "Image for container."

default = {

dev = "ghost:latest"

prod = "ghost:alpine"

}

}

-----------------------------------------------------------------------------------------------------

Convert container\_name to a map. Change the default to use key/value pairs. Set dev to blog\_dev and prod to blog\_prod.

variable "container\_name" {

type = map

description = "Name of the container."

default = {

dev = "blog\_dev"

prod = "blog\_prod"

}

}

---------------------------------------------------------------------------------------------------

Convert ext\_port to a map. Change the default to use key/value pairs. Set dev to 8080 and prod to 80.

variable "ext\_port" {

type = map

description = "External port for container."

default = {

dev = "8080"

prod = "80"

}

}

----------------------------------------------------------------------------------------------------

Now initialize Terraform.

PS C:\Users\user\Desktop\terraform\ass3\terraform-assignment-resources> terraform init

Initializing the backend...

Initializing provider plugins...

- Finding kreuzwerker/docker versions matching ">= 2.13.0"...

- Installing kreuzwerker/docker v2.15.0...

- Installed kreuzwerker/docker v2.15.0 (self-signed, key ID BD080C4571C6104C)

Partner and community providers are signed by their developers.

If you'd like to know more about provider signing, you can read about it here:

https://www.terraform.io/docs/cli/plugins/signing.html

Terraform has created a lock file .terraform.lock.hcl to record the provider

selections it made above. Include this file in your version control repository

so that Terraform can guarantee to make the same selections by default when

you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see

any changes that are required for your infrastructure. All Terraform commands

should now work.

If you ever set or change modules or backend configuration for Terraform,

rerun this command to reinitialize your working directory. If you forget, other

commands will detect it and remind you to do so if necessary.

------------------------------------------------------------------------------------------------------

Setup the Development environment

Create a workspace called dev.

Generate a Terraform plan. Output the plan and call it tfdev\_plan. Pass in a variable called env and set it to dev.

PS C:\Users\user\Desktop\terraform\ass3\terraform-assignment-resources> terraform plan -out="tfdevplan.tfplan"

var.env

env:dev or prod

Enter a value: dev

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:

+ create

Terraform will perform the following actions:

# docker\_container.container\_id will be created

+ resource "docker\_container" "container\_id" {

+ attach = false

+ bridge = (known after apply)

+ command = (known after apply)

+ container\_logs = (known after apply)

+ entrypoint = (known after apply)

+ env = (known after apply)

+ exit\_code = (known after apply)

+ gateway = (known after apply)

+ hostname = (known after apply)

+ id = (known after apply)

+ image = "ghost:latest"

+ init = (known after apply)

+ ip\_address = (known after apply)

+ ip\_prefix\_length = (known after apply)

+ ipc\_mode = (known after apply)

+ log\_driver = "json-file"

+ logs = false

+ must\_run = true

+ name = "blog\_dev"

+ network\_data = (known after apply)

+ read\_only = false

+ remove\_volumes = true

+ restart = "no"

+ rm = false

+ security\_opts = (known after apply)

+ shm\_size = (known after apply)

+ start = true

+ stdin\_open = false

+ tty = false

+ healthcheck {

+ interval = (known after apply)

+ retries = (known after apply)

+ start\_period = (known after apply)

+ test = (known after apply)

+ timeout = (known after apply)

}

+ labels {

+ label = (known after apply)

+ value = (known after apply)

}

+ ports {

+ external = 8080

+ internal = 2368

+ ip = "0.0.0.0"

+ protocol = "tcp"

}

}

# docker\_image.image\_id will be created

+ resource "docker\_image" "image\_id" {

+ id = (known after apply)

+ latest = (known after apply)

+ name = "ghost:latest"

+ output = (known after apply)

}

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:

+ container\_name = "blog\_dev"

+ ip\_address = (known after apply)

-----------------------------------------------------------------------------------------------------------------------------------------------------------

Saved the plan to: tfdevplan.tfplan

--------------------------------------------------------------------------------------------------------

Apply tfdev\_plan.

PS C:\Users\user\Desktop\terraform\ass3\terraform-assignment-resources> terraform apply tfdevplan.tfplan

docker\_image.image\_id: Creating...

docker\_image.image\_id: Creation complete after 0s [id=sha256:b05a58075ef6ad79016adb818a0a0b65097dfc58781e6eb6ab272fd32d976db9ghost:latest]

docker\_container.container\_id: Creating...

docker\_container.container\_id: Creation complete after 7s [id=e9abab9fea7c2aa8040009dee64b3ced58b06d4f486932b8a7fbe89f49fd841b]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

container\_name = "blog\_dev"

ip\_address = "172.17.0.2"

------------------------------------------------------------------------------------------------------

Setup the Production environment

Create a workspace called prod.

Generate a Terraform plan. Output the plan and call it tfprod\_plan. Pass in a variable called env and set it to prod.

PS C:\Users\user\Desktop\terraform\ass3\terraform-assignment-resources> terraform plan -out="tfprodplan.tfplan"

var.env

env:dev or prod

Enter a value: prod

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:

+ create

Terraform will perform the following actions:

# docker\_container.container\_id will be created

+ resource "docker\_container" "container\_id" {

+ attach = false

+ bridge = (known after apply)

+ command = (known after apply)

+ container\_logs = (known after apply)

+ entrypoint = (known after apply)

+ env = (known after apply)

+ exit\_code = (known after apply)

+ gateway = (known after apply)

+ hostname = (known after apply)

+ id = (known after apply)

+ image = "ghost:alpine"

+ init = (known after apply)

+ ip\_address = (known after apply)

+ ip\_prefix\_length = (known after apply)

+ ipc\_mode = (known after apply)

+ log\_driver = "json-file"

+ logs = false

+ must\_run = true

+ name = "blog\_prod"

+ network\_data = (known after apply)

+ read\_only = false

+ remove\_volumes = true

+ restart = "no"

+ rm = false

+ security\_opts = (known after apply)

+ shm\_size = (known after apply)

+ start = true

+ stdin\_open = false

+ tty = false

+ healthcheck {

+ interval = (known after apply)

+ retries = (known after apply)

+ start\_period = (known after apply)

+ test = (known after apply)

+ timeout = (known after apply)

}

+ labels {

+ label = (known after apply)

+ value = (known after apply)

}

+ ports {

+ external = 80

+ internal = 2368

+ ip = "0.0.0.0"

+ protocol = "tcp"

}

}

# docker\_image.image\_id will be created

+ resource "docker\_image" "image\_id" {

+ id = (known after apply)

+ latest = (known after apply)

+ name = "ghost:alpine"

+ output = (known after apply)

}

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:

+ container\_name = "blog\_prod"

+ ip\_address = (known after apply)

-----------------------------------------------------------------------------------------------------------------------------------------------------------

Saved the plan to: tfprodplan.tfplan

-------------------------------------------------------------------------------------------------------------

Apply tfprod\_plan.

PS C:\Users\user\Desktop\terraform\ass3\terraform-assignment-resources> terraform apply tfprodplan.tfplan

docker\_image.image\_id: Creating...

docker\_image.image\_id: Creation complete after 0s [id=sha256:cf0dc2177a49d260a09e2a47271d49f4d25b7daf9493abc2ffbe43d21f776c17ghost:alpine]

docker\_container.container\_id: Creating...

docker\_container.container\_id: Creation complete after 6s [id=20be98f5ea0d6c420c99faeef7e8051d5e0d9e0ef94561409e6fc1527aba0f11]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

container\_name = "blog\_prod"

ip\_address = "172.17.0.3"

--------------------------------------------------------------------------------------------------------------

Verify both environments work

Open a browser and navigate to the public IP. This should pull up the production environment.

Open a browser tab and navigate to the public IP on port 8080. This should pull up the development environment.