Stop etmiş olduğumuz makineyi start ederek remote ssh a bağlanıyoruz. Lesson 2 için main.tf in olduğu klasöre gidiyoruz.

Terraform plan ile kontrol edip terraform apply ile main.tf yi apply ediyoruz.

Ec2 umuz ve bucket ımız oluştu.

Terraform state list ile bunu görelim:

r@ip-172-31-22-176# terrafom-a.s:$ terraform state list 
• ec2-use 
instance. tf-ec2 
aws 
s3 bucket.tf-s3 
aws 
r@ip-172-31-22-176# terrafom-a.s:$ 
o ec2-use 

terraform validate dosyamızın için kontrol ediyor. hata var mı diye bakıyor ve geçerli bir dosya haline getiriyor. Örneğin bir value ismini yanlış girdik oradaki hatayı gösterir.

r@ip-172-31-22-176# terrafom-a.s:$ terraform validate 
• ec2-use 
Success! The configuration is valid. 

hata olmadığı için validated diyor.

apply dediğimizde de bu hatayı gösterir ancak çok uzun dosyalarda plan ve apply demeden önce validate diyebiliriz.

şimdi bilerek hata bırakalım:

v resource "aws instance" "tf-ec2" 
" ami-oe6dcf34cegesoe22" 
instance_type 
"t2.micro" 
key_n am 
"first-key" 
tags 
" Name" 
"tf-ec2" 
"aws s3 bucket" "tf-s3" 
v resource 
bucket 
" geovanni -tf-lesson" 

key-name in e sini sildik. 28. satırdaki süslü parantezi sildik ve terraform validate dedik:

r@ip-172-31-22-176# terrafom-a.s:$ terraform validate 
@ ec2-use 
Unclosed configuration block 
on main.tf line 24, in resource "aws s3 bucket" "tf-s3": 
24: resource "aws_s3_bucket" "tf-s3" { 
There is no closing brace for this block before the end of the file. 
file. 
This may be caused by incorrect brace nesting elseuhere in this 

bucket resource için kapama süslü parantezi yok diyor. Onu düzeltip kaydettik ve tekrar terraform validate dedik:

r@ip-172-31-22-176# terrafom-a.s:$ terraform validate 
@ ec2-use 
Unsupported argunent 
on main.tf line 
18: key 
An argument named 
18, in resource "aws instance" "tf-ec2": 
"first-key" 
"key_rwn" is not expected here. 
Did you mean 
"key_rwne" 

bu kez de key\_nam de hata olduğunu gösteriyor ve key\_name mi demek istediniz diyor. onu da düzeltip kaydetiyoruz.

Hataları tek tek veriyor validate komut.

terraform fmt (format) komutumuz var. dosyada girintiler, boşluklarla ilgili bir sorun olduğu durumlarda;

resource 
key_name 
"Name" — 
"aws instance" 
"tf-ec2" 
" ami -øe6dcf34coge5øø22" 
instance_type = ' 
"oliver" 
tags { 
"tf-ecT' 
resource "aws s3 bucket" 
_ micro" 
"tf-s3" 
bucket — "oliver•-tf-test-bucket-Iesscn2" 

terraform fmt bu boşluk hatalarını otomatik düzeltir. aynı dizin içindeki bütün .tf uzantılı dosyaları düzeltir.

terraform validate te yine aynı dizin içindeki bütün .tf uzantılı dosyalardaki hataları gösterir.

resource "aws instance" "tf—ec2" { 
" ami -øØ6dcfgac " 
instance_type - 
micro" 
— "cliver" 
key_na me 
taes - 
"Name " 
"tf• ec2 

terraform console komutumuz bize ekstra bir konsol açar.

terraform console 
O ec2-use 
1 

şimdi önce echo "hello devops" > cloud ile bir dosya oluşturalım.

r@ip-172-31-22-176# terrafom-a.s:$ terraform console 
o ec2-use 
> aws instance.tf-ec2 
" arn" 
"ami-øø6dcf34cøgesøø22" 
" arn : aws : ec2 : us-east-I : 995714465218 : instance/i-ø4bc7727gbba28aø1" 
"associate public_ip_address" 
true 
" availability _ zone" 
"us-east-Id" 
" capacity_reservation_specification" 
"capacity_reservation_preference" 
= tolist([ 
"open" 
"capacity_reservation_target" = tolist([]) 

terraform console ile bize verdiği terminal ile instance on bütün bilgilerini çektik. daha spesifik bilgiler ide çekebiliriz:

> aws_instance. tf-ec2. private_ip 
"172.31.27.43" 

instance ın private\_ip sini çektik.

Bunlar dışında terraform içinde fonksiyonlar var. Terraforma gömülü belli başlı fonksiyonları kullanabiliyoruz.

> min 

min (1,2,3) diyerek rakamların en küçüğünü döndürüyor.

011 乚 JamOl 《 
„OIIeu„ 

(büyük harfleri küçük harflere çeviriyor.

> file("${path.module}/cloud") 
hello devops 

module içindeki dosyayı getirebiliyor. main.tf i de getirirdi. aynı dizinde çalışıyorsak ${path.module}/ yazmasak da olur.

> aws s3 bucket.tf-s3 
"acceleration status" 
"acl" 
" arn" 
= tostring(null) 
"arn : aws : s3: : :geovanni-tf-lesson" 
"bucket" 
"geovanni-tf-lesson" 
"bucket domain name" 
" geovanni-tf-lesson. s3. . com" 
"bucket prefix" = tostring(null) 
" n_name " 
" geovanni-tf-lesson. s3. . com" 
"cors rule" 
= tolist([]) 
"force_destroy" = false 
"grant" = toset([ 

s3 bucket ı da çekebiliyoruz.

bucket ismini çekelim:

> aws s3 bucket.tf-s3. bucket 
"geovanni-tf-lesson" 

bunları eşitli nedenlerle terraform içinde kullanıyoruz. burada örnek olsun diye yazdık.

exit ya da ctrl + c ile bu terminalden çıkabiliriz.

terraform show komutumuz var, tfstate i daha okunaklı bir halde getirir:

r@ip-172-31-22-176# terrafom-a.s:$ terraform show 
• ec2-use 
# aws instance.tf-ec2: 
resource "aws_instance" "tf-ec2" { 
arn 
associate public_ip_address 
availability _ zone 
cpu_core_count 
cpu_threads_per_core 
disable_api_stop 
disable_api_termination 
ebs_optimized 
get _ password_data 
hibernation 
"ami-øø6dcf34cøgesøø22" 
" arn : aws :ec2 : us-east-I : 995714 
true 
" us-east-Id" 
false 
false 
false 
false 
false 

terraform show, terraform console komutları tfstate dosyasını kullanır.

terraform graph komutumuz var. resource ların birbirine bağlantısı varsa onları şema gibi gösterir svg formatında.

bunun için bu yüklemeleri yapmaya gerek yok <https://dreampuf.github.io/GraphvizOnline>

sitesinden yapabiliyoruz.

terraform graph komutunu giriyoruz

r@ip-172-31-22-176# terrafom-a.s:$ terraform graph 
• ec2-use 
digraph { 
compound = "true" 
newrank 
"true" 
subgraph "root" 
" [root] 
" [root] 
" [root] 
" [root] 
" [root] 
" [root] 
" [root] 
" [root] 
(expand)" 
provider[\"registry.terraform. io/hashicorp/aws\"]" [label 
aws instance. tf-ec2 
aws s3 bucket.tf-s3 
aws instance. tf-ec2 
aws s3 bucket.tf-s3 
(expand)" [label = "aws_instance.tf-ec2", 
[label = "aws_s3_bucket.tf-s3", 
shape = "box"] 
shape = "box"] 
" provider [ \ " registry. terraform. io/hashico 
(expand)" "[root] io/hashicorp/aws\"]" 
(expand)" "[root] io/hashicorp/aws\"]" 
io/hashicorp/aws\"] (close)" "[root] aws_instance.tf-ec2 (expand)" 
io/hashicorp/aws\"] (close)" "[root] aws_s3_bucket.tf-s3 (expand)" 
root" "[root] io/hashicorp/aws\"] (close)" 

verdiği çıktıyı yukarıdaki siteye yapıştırıp bir grafik görebiliriyoruz:

compound 
— "true" 
newrank = 
"true" 
subgraph "root 
(expand) " 
(expand)" 
. te r rafom 
. terrafom 
. terrafom 
[label 
[label 
Engine : 
— "aws instance.tf-ec2" 
shape 
— • 'aws s3 bucket.tf-s3% 
shape 
"provider [V 
[label 
dot 
F ormat: 
14 
15 
Cl Show raw output 
[root] root 
Downloadl 
"Croot] 
"[root] 
"Croot] 
"Croot] 
"Croot] 
"[root] 
"Croot] 
"Croot] 
aws instance .tf-ec2 
aws s3 bucket. tf-s3 
provider C \ "registry 
aws instance .tf-ec2 
aws s3 bucket. tf-s3 
provider C \ registry 
provider( 
\ registry 
(expand)" - > " [root] provider 
(expand)" - > " [root] provider 
. io/ hashicorp/aws\ 
"J (close)" 
"[root) a: 
. io/ hashicorp/aws\ 
"J (close)" 
"[root) a.' 
root" -> "[root] (clcs 
[root] (close) 
aws instance_tf-ec2 
aws s3 bucket_tf-s3 
terraform_ioihashicorp/aws"] 
RESET 

genelde main.tf variabşe.tf ve output.tf olarak ayrılır dosyalar.



output dediğimizde bizden isim vermemizi ister. bu isim referans verirken kullanacağımız isimdir. altında value = diyerek neyi çekeceğini giriyoruz:

6) READMEmd •r 
outputtf X 
terraform-lesson > terraform-aws > output.tf > output "tf_example_s3_l 
"tf_example_public_ip" { 
output 
value = 
aws_instance. tf -ec2. publ ic_ip 
"tf_examp1e_s3 meta 
output 
value = 
aws_s3 bucket. tf-s3.region 

biz terraform apply dediğimizde outputları bize çekecek artık.

r@ip-172-31-22-176# terrafom-a.s:$ terraform apply 
o ec2-use 
aws_instance.tf-ec2: Refreshing state... 
s3_bucket.tf-s3: Refreshing state... [id=geovanni-tf-lesson] 
Changes to Outputs: 
tf_examp1e_s3_meta 
"54.242.182.282" 
"us-east-I" 
You can apply this plan to save these new output values to the Terraform state, 
Do want to perfom these actions? 
Terraform will perform the actions described above. 
Only •yes' will be accepted to approve. 
Enter a value: 
without c 

output u değiştirecem diye belirtiyor.

Changes to Outputs: 
"54.242.182.282" 
"us-east-I" 
You can apply this plan to save these new output values to 
Do want to perfom these actions? 
Terraform will perform the actions described above. 
Only •yes • will be accepted to approve. 
Enter a value: yes 
Apply canplete! Resources: added, 
changed, 
destroyed. 
Outputs: 
tf_example_ 
s3 meta 
"54.242.182.282" 
"us-east-I" 

en altta outputları verdi. tfstate içinde output boşken orada da yer alıyor artık:

READMEmd outputtf X 0 
terraform.ffstate X 
terraform-lesson > terraform-aws > terraform.tfstate > 
"version": 4, 
"terraform version : 
" "1.3.g", 
"serial": 12, 
"lineage": "4c216e94-ødd8-cefd-b3fe-e7a7dß4edd69" , 
"outputs" : 
" ic_ip " • 
"type": "string" 
"value" : 
" 54.242.182.2e2", 
"tf_examp1e_s3 meta" : 
"value": 
"us-east- " 

terraform output ile bu çıktıyı da görebiliyoruz:

• ec2-use 
reip-172-31-28 
3_meta — 
terra-form-æs:$ terraform output 
"my public ip is 35.153-51.36" 
us-east-I" 

tek bir output u görmek istiyorsak onu da çekeriz:

r@ip-172-31-22-176# terrafom-a.s:$ terraform output 
• ec2-use 
"54.242.182.282" 

-json formatınd da output çekebilriiz:

terra-form-was: $ terraform output 
• ec2-use 
"t f_ex c _ip " : 
"sensitive": false, 
"type" : 
"string" , 
"value": "my public ip is 35.153.51.36" 
"tf_exampl e_s3_meta : 
"sensitive": false, 
"type" 
" string" , 
"value": "us-east-I" 
- j son 

terraformda output ları baştan düşünmemiz lazım, bir yere çekip yazmamız gereken bilgileri düşünüp configuration a eklememiz gerekir.

eğer terraform dışında konsol üzerinde bir değişiklik yapıldığında refresh ederek terraform gerçek durumla tfstate i karşılaştırıyor ve tfstate i düzeltiyor.

terraform state list ile resource ları görelim, iki resource umuz var. s3 bucket ı konsol üzerinden silelim.

terraform state list i terkar girersek s3 varmış gibi çıktı verecek:

r@ip-172-31-22-176# terrafom-a.s:$ terraform state list 
• ec2-use 
aws instance. tf-ec2 
aws s3 bucket.tf-s3 

çünkü terraform tfstate e bakarak bu çıktıyı veriyor. illa bir şeyin silinmesine gerek yok konsol üzerinden byapılan başka işlemler için de terraform apply -refresh-only komutunu giriyoruz:

r@ip-172-31-22-176# terrafom-a.s:$ terraform apply -refresh-only 
o ec2-use 
aws_instance.tf-ec2: Refreshing state... 
s3_bucket.tf-s3: Refreshing state... [id=geovanni-tf-lesson] 
Note: Objects have changed outside of Terrafom 
Terraform detected the following 
changes made outside of Terraform since the last "ter 
# æs s3 bucket.tf-s3 has been 
deleted 
resource "aws s3 bucket" "tf-s3 { 
"arn : aws : s3: : :geovanni-tf-lesson" 
null 
arn 
- bucket 
bucket domain rwne 
"geovanni-tf-lesson" 
null 
" geovanni-tf- lesson. s3. . com" 
"geovanni-tf-lesson.s3.amazonaws.com" 
false 
null 
"Z3AQBSTGFYJSTF" 
> null 
null 
null 
bucket_regional_domain_name 
force_destroy 
hosted zone id 

terraform dışında değişiklikler olduğunu belirtiyor.

output ta yapılacak değişiklikleri söylüyor ve yes girerek confirm etmemizi istiyor:

Changes to Outputs: 
tf_examp1e_s3_meta 
"us-east-I" (known after apply) 
You can apply this plan to save these new output values to the Terr 
Would like to update the Terrafom state to reflect these detecte( 
Terraform will write these changes to the state without modifying ar 
There is no undo. Only •yes' will be accepted to confirm. 
Enter a value: yes' 

confirm ettik ve tfstate i güncelledi. şimdi terraform apply -auto-approve dersek ne olacak?

.tf dosyalarımızın içeriğini uygulayacak, tfstate bakacak orada bucket yok, bucket oluşturacak:

Apply canplete! Resources: 1 added, 
Outputs: 
changed, 
destroyed. 
= "54.242.182.282" 
tf_examp1e_s3_meta 
"us-east-I" 
terrafomn-a.s: 
ec2-use 

s3 bucket ımız oluştu.

şimdi main.tf te değişiklikler yapıyoruz. eski ami yi commentleyip yeni bir ami ekliyoruz. name i değiştiriyoruz ve eski bucket ı comment yapıp yeni bir bucket ismi veriyoruz.

terraform apply -auto-approve komutunu giriyoruz:

Apply canplete! Resources: 2 added, 
"52.72.18.118" 
changed, 
2 destroyed. 
Outputs: 
tf_example_ 
s3 meta 
"us-east-I" 

ami değiştiği için ec2 yu destroy etti ve yeni ami ile ubuntu oluşturdu. bucket ı destroy etti ve yeni bucket oluşturdu.

terraform apply -refresh=false

komutu da refresh yapma demektir.

v output "tf_example_private_ip" 
value = aws_instance.tf-ec2.private_ip 

yeni bir output ekledik ve refresh=false komutunu girelim:

Apply canplete! Resources: added, 
tf_example_public_ip 
changed, 
destroyed. 
Outputs: 
tf_example_ 
"172.31.52.132" 
"52.72.18.118" 
"us-east-I" 
s3 meta 

son eklediğimiz output u verdi ve tfstate i güncellemedi ancak bu tavsiye edilmez çünkü konsolda yapılan bir değişiklik tfstate e işlenmemiş olur.

şimdi terraform destroy diyoruz.

gerçek hayatta;

resource "aws instance" "tf-ec2" { 
ami 
instance_type "t2. micro" 
" ami " 
" oliver 
key_name 
tags = 
resource 
bucket = 
"Name" = "tf-ec2-ubuntu" 
"aws_s3_bucket" "tf-s-3" { 
"oliver-tf-test-bucket -les son2-new" 

bu şekilde karşısında ami, type direk yazmaz. bunlar hep variable lara atanmıştır. dolayısıyla .tf dosyaları module gibi kullanılır. yani başkaları da variable ları değiştirerek bu terraform dosyalarını kullanabilir.

şimdi main.tf içine ya da ayrı bir variable.tf dosyası oluşturarak variable ekleyelim:

terraform-lesson > terraform-aws > variable.tf > variable "ecz 
variable "ec2 name" 
default 
"emre-ec2" 
description 
"it describes ec2's name" 

ec2 name için oluşturduk bunu. variable yazdığımızda eklenti bize yazış formunu getirecektir. default = diyerek değerin karşılığını yazıyoruz. ec2 oluştururken name vermezsek defaultu budur.

variable 
default 
variable 
default 
"ec2_type" { 
"t2. micro" 
"ec2 ami" 
" ami-ß742b4e673e72ß66f" 

ami ve type ı da variable olarak ekledik.

şimdi resource ta referans gösterecez:

karşısına gelip var. yazdığımızda tamamlayacaktır:

resource 
#ami = 
"aws 
instance" "tf-ec2" 
" aml- 
• ee6dcf34cegesee22" 
var.ec2 ami 
instance_type = v 
key_name = 
"first. var.ec2 ami 
tags 
var. ec2 name 
*"Name" 
"tf-e var.ec2_type 
"tf-ec2- vare 
Name 
varm 

Name i ${} içinde yazdık:

resource 
#ami = 
"aws instance" "tf-ec2" 
" ami -ee6dcf34cegesee22" 
var.ec2 ami 
instance_type = var. ec2_type 
key_name = 
tags 
*"Name" 
Name 
"first-key" 
"tf-ec2" 
${var. ec2_name} -instance " 

bucket ismi için de variable atayalım:

resource 
#bucket 
bucket 
"aws s3 bucket" "tf-s3" 
"geovanni-tf-lesson" 
var. bucket name 
default 
"ami 
14 
variable "bucket_name" { 
default 
" geovanni - tf -bucket 

output ta bucket output unu da .region u .bucket olarak güncelleyelim:

output "tf_examp1e_s3 meta 
value = aws s3 bucket. tf-s3.bucket 

şimdi terraform apply diyelim:

Apply canplete! Resources: 2 added, changed, 
Outputs: 
"172.31.28.6" 
tf_example_public_ip = "54.226.68.232" 
tf_examp1e_s3_meta = "geovanni-tf-bucket" 
terraform-m.s: 
ec2-use 
destroyed. 

resource larımızı oluşturdu ve outputumuzu verdi.

gerçek ayakta provider lar, variable lar gibi bloklar için de ayrı dosyalar oluşturulur. şimdi de variable.tf oluşturup variable ları oraya atıyoruz.

şimdi bucket name olan variable ı commentleyelim:

variable 
"bucket name" 
" geovanni-tf-bucket" 

bucket name olmadan terraform plan dersek ne olur? terraform terminalde bunu bizden ister:

@p-172-31 
o ec2-use 
var.s3 bucket name 
Enter a value: 
-28-1# terraform-aws:$ 
terraform plan 

burada isim vermemiz gerekir. bucket name i farklı girdiğimiz için önceki silip yenisini oluşturacağını söyledi:

Plan: 1 to add, to change, 1 to destroy. 
Changes to Outputs: 
tf_examp1e_s3_meta 
"geovanni-tf-bucket" 
"geovanni-bucket" 

şimdi -var flag ini kullanacağız:

r@ip-172-31-22-176# terrafom-a.s:$ terraform plan 
o ec2-use 

bu komut yani -var flagi kullanarak komut satırında bu variable ı tanımlayabiliriz.

Plan: 1 to add, to change, 1 to destroy. 
Changes to Outputs: 
tf_examp1e_s3_meta 
"geovanni-tf-bucket" 
"geovanni-new-bucket" 

export komutuyla da variable tanımlayabiliriz:

export TF\_VAR\_s3\_bucket\_name=oliver-env-varible-bucket

komutu ile:

r@ip-172-31-22-176# terrafom-a.s:$ export TF VAR 
• ec2-use 
r@ip-172-31-22-176# terrafom-a.s:$ terraform plan 
o ec2-use 

daha sonra yine terraform plan dediğimizde bunu da görecektir:

Plan: 1 to add, to change, 1 to destroy. 
Changes to Outputs: 
tf_examp1e_s3_meta 
"geovanni-tf-bucket" 
"geovanni-env-varible-bucket" 

bir diğer yöntem de aynı dizinde terraform.tfvar dosyası oluşturmaktır. bu dosyanın oluşturulabilmesi için variables dosyasının olması gerekir. variables.tf içindeki varible bloğundaki adı tfvars dosyasının içine yazarak değer veriyoruz:

dosyanın içine yapıştıralım:

s3\_bucket\_name = "tfvars-bucket"

terraform.tfvars dosyası varsa terraform bu dosyadan variable çeker:

Plan: 1 to add, to change, 1 to destroy. 
Changes to Outputs: 
tf_examp1e_s3_meta 
"geovanni-tf-bucket" "tfvars-bucket" 

s3\_bucket\_name i tfvars-bucket olarak güncelleyeceğini belirtti.

variables ta önemli olan dosyaların kullanışlı olması. şirket içinde ana dosya düzenine dokunulmadan başkaları da oluşturduğumuz dosyaları nasıl kullanabilir? bunu arıyoruz.

şirketin farklı departmanları farklı .tfvars dosyaları oluşturabilirler:

istediğimiz isimle bir .tfvars dosyası oluşturalım:

D README.md terraform.tfvars geovanni.tfvars X 
terraform-lesson > terraform-aws > geovanni.tfvars > @ s3_bucket_name 
s3 bucket_name = "geovannil-tfvar-bucket" 

onun içine de s3\_bucket\_name karşılığında bir bucket name verdik.

peki farklı iki .tfvars dosyamız var terraform plan dersek hangisini kullanır. terraform.tfvars ı kullanır.

terraform plan --var-file="geovanni.tfvars

bu komutu girersek bu kez geovanni.tfvars dosyasını dikkate alacaktır.

Plan: 1 to add, to change, 1 to destroy. 
Changes to Outputs: 
tf_examp1e_s3_meta 
"geovanni-tf-bucket" 
"geovanni-tfvar-bucket" 

bir diğer tfvars dosyası yöntemi de auto.tfvars dosyası oluşturmaktır. eğer auto.tfvars dosyası varsa onu dikkate alır.

READMEmd 
geovanni.auto.tfvars X terraform.tfi 
terraform-lesson > terraform-aws > geovanni.auto.tfvars > @ s3 
s3 bucket name 
"oliver-auto-tfvar-bucket" 

isim.auto.tfvars dosyası oluşturduk ve içine bucket name yazdık. terraform plan dedikten sonra bakalım:

Plan: 1 to add, to change, 1 to destroy. 
Changes to Outputs: 
tf_examp1e_s3_meta 
"geovanni-tf-bucket" "oliver-auto-tfvar-bucket" 

sadece terraform plan kodu ile auto.tfvars dosyasını dikkate aldı.

Any -var and -var-file options on the command 
line, in the order they are provided. 
Any *.auto.tfvars or * .auto.tfvars.json files, 
processed in lexical order of their filenames. 
The terraform. tfvars.json file, if present. 
The terraform. tfvars file, if present. 
Environment variables 

komut satırında -var kullanıldıysa önce onu dikkate alır. yoksa auto.tfvars ya da auto.tfvars.json dosyalarını dikkate alır.

onlar yoksa terraform.tfvars.json dikkate alınır. O yoksa terraform.tfvars dikkate alınır. Bunlardan sonra da environment variables dikkate alınır.

main.tf içinde sık kullanılan bir value için "locals" özelliği kullanılır.

örneğin bütün resource larda aynı tag kullanıyoruz, onu locals ile gösterebiliriz:

v locals { 
mytag 
"emre- local -name " 

bunu resource ta şu şekilde gösteririz:

resource 
#ami = 
"aws instance" "tf-ec2" 
" ami -ee6dcf34cegesee22" 
var.ec2 ami 
instance_type = var. ec2_type 
key_name = 
tags 
*"Name" 
Name 
"first-key" 
"tf-ec2" 
"${local .mytag} -come from locals" 

instance ımızın name tagi "emre-local-name" olacak.

bucket a da tag ekleyelim:

resource "aws s3 bucket" "tf-s3" 
#bucket 
"geovanni-tf-lesson" 
bucket = var .s3 bucket name 
tags 
" Name" 
${local.mytag}-come from local*" 

local, başkaları tarafından değiştirlme ihityacı olmayacak ve main.tf içinde çok kullanılan değerler için kullanılır.

terraform plan diyelim:

# æs s3 bucket.tf-s3 will be updated in-place 
" aws 
resource 
id 
tags 
"Name" 
tags_all 
"Name" 
bucket" "tf-s3" { 
"geovanni-auto-tfvar-bucket" 
"emre-local-name-come from locals" 
"emre-local-name-come from locals" 
unchanged attributes hidden) 
unchanged blocks hidden) 
Plan: to add, 2 to change, to destroy. 

tag ekleneceğini belirtti. terraform apply dediğimizde tag leri ekleyecek.

Outputs: 
tf_example_ 
s3 meta 
r@ip-172-31-22 
) ec2-use 
Plan: to add, 2 to 
change, to destroy. 
instance.tf-ec2: 
modifying... 
s3 bucket.tf-s3: 
modifying... [id=geovanni-auto-tfvar-bucket] 
s3 bucket.tf-s3: 
modifications complete after øs [id=geovanni-auto-tfvar-bucket] 
æs instance.tf-ec2: modifications complete after Is [id=i-Ü69fe3cø7874d9d] 
Apply canplete! Resources: added, 2 changed, destroyed. 
tf_example_public_ip 
"172.31.28.6" 
"54.226.68.232" 
"geovanni-auto-tfvar-bucket" 
-176# terraform-m.s : 

resource local name ini değiştirerek aynı kod bloğuyla fazla resource oluşturlabilir:

resource " aws instance 
"tf-ec2" 
ami 
= var. ec2 ami 
instance type = var. ec2_type 
key_name 
tags = 
"Name" 
= "oliver" 
"$1 loc al. -instance " 
resource "aws_instance" "e-ec2-x { 
ami 
= var. ec2 ami 
instance_type var. ec2_type 
key_name 
tags = 
"Name" 
— "oliverl' 
"${loc al. mytag]• -instance " 

ancak terraformda bunun yolu bu değil.

bu count ile yapılır.

resource 
#ami = 
"aws 
instance" "tf-ec2" 
" aml- 
• ee6dcf34cegesee22" 
var.ec2 ami 
instance_type = var. ec2_type 
key_name = 
"first-key" 
count 
tags 
*"Name" 
Name 
"tf-ec2" 
"${local .mytag} -come from locals" 

ancak biz bunu direk yazmayacaz variable olarak ekleyecez. variables.tf ye gidip num\_of\_bucket variable ekleyelim:

variable "num of buckets" 
default = 2 

resource 
#bucket 
bucket 
"aws s3 bucket" "tf-s3" 
"geovanni-tf-lesson" 
var .s3 bucket name 
count = var. num of buckets 
tags 
" Name" 
${local.mytag}-come from locals" 

main.tf de bucket resource altına da count ile bu variable a referans verdik.

ancak AWS iki bucket a izin vermez çünkü bunlar aynı isimle oluşamaz AWS de. terraform hata vermez ama iki tane oluşturamaz.

"aws s3 bucket" "tf-s3" 
resource 
+ 11}" 
${var. s3_bucket_name 
bucket = " 
count var .num of buckets 

bucket ismini yukarıdaki şekilde girersek count.index diye terraform resource a index veriyor bu şekilde eklediğimizde bucket isimlerine o index numarasını veriyor. uniqe liği bu şekilde sağladık. +1 yazmamızın sebebi 0 dan başlamaması.

output u da güncellemeliyiz çünkü bucket ismini output olarak istemiştik:

output "tf_examp1e_s3_meta" { 
value = aws s3 bucket. tf-s3.*.bucket 

artık terraform apply diyebiliriz:

Apply canplete! Resources: 2 added, 
Outputs: 
"172.31.28.6" 
tf_example_public_ip 
"54.226.68.232" 
tf_examp1e_s3_meta = 
"geovanni-auto-tfvar-bucket-l" , 
" geovanni- auto-tfvar- bucket- 2" , 
changed, 
1 destroyed. 

...bucket-1

...bucket-2 olarak eklediğini belirtiyor.

Conditional Expressions:

<https://developer.hashicorp.com/terraform/language/expressions/conditionals>

bu bir programlama dili değil, yaratıcılık gerektirmez, yukarıdaki linkten dokümantasyondan alıyoruz.

Syntax 
The syntax Of a conditional expression is as follows: 
condition true val : false val 
ILLL9y 

bizim örneğimize:

count = var.num\_of\_buckets != 0 ? var.num\_of\_buckets : 3

count = var.num\_of\_buckets 0 a eşit değilse var.num\_of\_buckets kadar oluştur, 0 a eşitse 3 tane oluştur.

resource ta bucket count karşısına bunu yazıyoruz ve terraform apply diyoruz:

No changes. Your infrastructure matches the configuration. 
Terraform has ccnpared your real infrastructure against your config 
Apply canplete! Resources: added, changed, destroyed. 
Outputs: 
tf_example_public_ip 
tf_examp1e_s3_meta = 
"172.31.28.6" 
"54.226.68.232" 
"geovanni-auto-tfvar-bucket-l" , 
" geovanni- auto-tfvar- bucket-2" , 
r@ip-172-31-22-176# terrafomn-a.s : 
o ec2-use 

değişiklik yok çünkü var.num\_of\_bucket 0 a eşit değil, altyapınız conf ile uyumlu diyor.

şimdi for\_each a bakalım:

<https://developer.hashicorp.com/terraform/language/meta-arguments/for_each>

Şimdi IAM role a gidip rolümüze IAMFullAccess i ekleyelim.

users variable larını ekleyelim:

variable "users" {

default = ["santino", "michael", "fredo"]

}

şimdi IAM user ekleyecek bir resource bloğunu main.tf e ekleyelim:

resource "aws\_iam\_user" "new\_users" {

for\_each = toset(var.users)

name = each.value

}

toset terraform da gömülü bir fonksiyondur, listeyi set e çevirir.

users variable ını set e çevir diyoruz.

oluşturacağımız bu userlar için de bucket oluşturmak istiyoruz. count yerine for\_each kullanacağız:

"aws s3 bucket" "tf-s3" 
resource 
#bucket 
"geovanni-tf-lesson" 
#bucket 
"${var. count . index 
var. num of buckets e ? var. num of 
#count - 
for each 
toset(var. users) 
bucket 
" emre-tf-s3-bucket-${each. val ue}" 
tags 
" Name" 
${local.mytag}-come from locals" 

count ile 3 tane bucket oluşturmadık da her user için bir bucket oluşturarak 3 tane bucket oluşturmuş olacağız. bucket = each.value olarak da bırakabilirdik ancak isimlerin başına emre-tf-s3-bucket eklemesi için böyle yazdık.

output a da bir ekleme yapalım:

output "uppercase\_users" {

value = [for user in var.users : upper(user) if length(user) > 6]

}

outputta user ların ismini döndürmek istiyoruz ve for döngüsü kullanıyoruz. terraformda gömülü bir döngü bu. var.users içinde olan her user için upper hale getir. upper da terraform da gömülü, büyük harflere dönüştür demek. if diyerek şart ekliyoruz ve length, yani ismin uzunluğu, 6 dan büyükse .

Sonuç olarak; var.users içindeki her user için; uzunluğu 6 dan büyük olan isimleri büyük harfle output olarak döndür.

for döngüsü linki:

<https://developer.hashicorp.com/terraform/language/expressions/for>

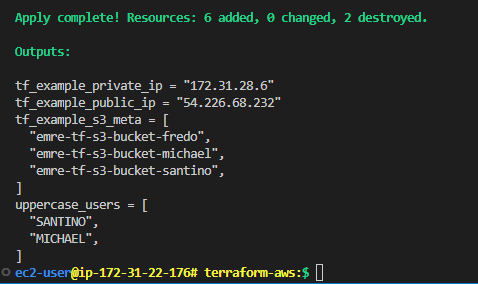
bucket ta count u sildiğimiz için outputa da değişiklik yapmamız gerekiyor:

output "tf\_example\_s3\_meta" {

value = values(aws\_s3\_bucket.tf-s3)[\*].bucket

şu anda apply etmeye hazırız.

output ta userları nasıl verdiğini de görüyoruz:



Artık terraform destroy diyebiliriz.