**Terraform state mv**

這邊用範例講解，同時注意三個部分的差異程度

* .tf 中描述的 resource block，也就是依據業務需求想達到的 desired state
* 遠端公有雲上對應的 resource infrastructure，也就是 actual state
* terraform state 中存在 resource，也就是 metadata，或是中間產物

terraform 是協助工程師管理上述三組 state，達到狀態統一

terraform state 變更 (ex. mv 等) 只管理 terraform state 這一部分，.tf resource 與 remote resource 是維持原狀的

state 變更會在三組 state 中產生分歧，這些分歧會產生額外的效果，這是我們在 state manipulation 時須要格外注意的

**Terraform state mv**

<https://www.terraform.io/docs/cli/commands/state/mv.html>

**根據官網內容 state mv 做以下事情**

* terraform state mv resource 到新的 address

一樣回到 azure/foundation/compute\_network 的範例

cd azure/foundation/compute\_network

terragrunt state list

module.network.data.azurerm\_resource\_group.network

module.network.azurerm\_subnet.subnet[0]

module.network.azurerm\_subnet.subnet[1]

module.network.azurerm\_subnet.subnet[2]

module.network.azurerm\_virtual\_network.vnet

**rename .tf & state mv**

例如：我們因為業務需求變更，希望 rename module.network 的 module name，從 module.network 變成 module.private-network

* 這邊只是舉個範例，為改而改

git diff

*# https://github.com/Azure/terraform-azurerm-network*

-module "network" {

+module "private-network" {

source = "Azure/network/azurerm"

resource\_group\_name = var.resource\_group\_name

(END)

改完之後，回到 azure/foundation/compute\_network ，進行 terraform plan，請問會看到什麼變化？這邊大家依據過去所學，預測一下 terraform 的行為

這邊給幾個提示

* module 是 terraform 的基本單元，rename module 會有什麼影響？

cd azure/foundation/compute\_network

terragtunt plan

╷

│ Error: Module not installed

│

│ on compute\_network.tf line 2:

│ 2: module "private-network" {

│

│ This module is not yet installed. Run "terraform init" to install all

│ modules required by this configuration.

╵

ERRO[0015] 1 error occurred:

\* exit status 1

跳出 module not installed 的錯誤，於是這邊我們進行 initA

* 效果是新稱 module.private-network
* 複習可以翻回去前面的第？章

terragrunt init

Initializing modules...

Downloading Azure/network/azurerm 3.5.0 for priavte-network...

- priavte-network in .terraform/modules/private-network

Initializing the backend...

Initializing provider plugins...

於是在進行 plan

* 如果 output 中友引用原來的 module，會需要跟著修正 output

terragrunt plan

Terraform used the selected providers to generate the following execution

plan. Resource actions are indicated with the following symbols:

+ create

- destroy

Terraform will perform the following actions:

*# module.network.azurerm\_subnet.subnet[0] will be destroyed*

}

*# module.network.azurerm\_subnet.subnet[1] will be destroyed*

}

*# module.network.azurerm\_subnet.subnet[2] will be destroyed*

}

*# module.network.azurerm\_virtual\_network.vnet will be destroyed*

}

*# module.private-network.azurerm\_subnet.subnet[0] will be created*

}

*# module.private-network.azurerm\_subnet.subnet[1] will be created*

}

*# module.private-network.azurerm\_subnet.subnet[2] will be created*

}

*# module.private-network.azurerm\_virtual\_network.vnet will be created*

}

Plan: 4 to add, 0 to change, 4 to destroy.

Changes to Outputs:

~ vnet\_id = "/subscriptions/.../resourceGroups/terraform-30-days/providers/Microsoft.Network/virtualNetworks/acctvnet" -> (known after apply)

~ vnet\_subnets = [

...

+ (known after apply),

+ (known after apply),

+ (known after apply),

]

plan 的結果，獲得 4 to add, 4 to destroy，terraform 認為要移除遠端已經存在的 network，並重新建立新的

terraform 為何會有這樣的判斷？

* .tf 中，舊的 module.network 已經不復存在，而 remote infra 跟 state 中都存在，所以 terraform 認為需要 destroy
* .tf 中，多產生一個新的 module.private-network，state 中並不存在，所以 terraform 認為需要 create

完全符合之前討論過 terraform 運作的邏輯

然而這樣符合我們的需求嗎？

實務上，我們盡量避免 rename resource / module，可以避免上述情形時常發生，然而這邊的 state manipulation 就是在講非常情形：我們真的被迫要做 module rename，然而不希望遠端被 destroy + create

* 也許遠端 subnet 已經正在使用，destroy subnet 會影響其他服務
* terraform module name 在 azure cloud 上其實是無意義的
  + module / resource name 只是用在 terraform .tf 與 state 中做 resource addressing 的路徑名稱
  + 對於 azure cloud 而言，resource / module 都沒有意義，有意義的事 subnet 本身
  + 因為管理用的 resource addressing 變更，而影響到實際 remote resource 這點，可說是 terraform 的限制，或是說 terraform 本身就不是設計來做這類操作的

**Fix rename with state mv**

在回憶一下三個部分

* .tf 被我們 rename module.network -> module.private-network
* state 仍然是原先的 module.network
* remote 透過 state 中的 module.network 的 metadata 做對應

我們只要能夠 rename state 中的 module.network 變成 state module.private-network

* state module. 的內容完全不變，但是名稱變成 module.private-network
* .tf 已經是 module.private-network，這樣就能符合 state 內容
* 由於 state module.private-network 的 metadata 與 module.network 完全一樣，因此仍然是對應到原先的 azure cloud resource

於是我們進行 state mv，嘗試看看

* 養成能夠 dry-run 就 dry-run 的好習慣，畢竟這邊有 .tf 變更
* 如果 .tf 寫錯，mv 也搬錯 address 之後要在搬回來很麻煩

terragrunt state mv --dry-run SOURCE DESTINATION

terragrunt state mv --dry-run module.network.data.azurerm\_resource\_group.network module.private-network.data.azurerm\_resource\_group.network

Would move "module.network.data.azurerm\_resource\_group.network" to "module.private-network.data.azurerm\_resource\_group.network"

看起來跟我們預想的狀況相符，於是我們為 module 底下所有 resource 進行 state mv

* 記得 [0] 在 bash 中需要 quote

terragrunt state mv module.network.data.azurerm\_resource\_group.network module.private-network.data.azurerm\_resource\_group.network

terragrunt state mv "module.network.azurerm\_subnet.subnet[0]" "module.private-network.azurerm\_subnet.subnet[0]"

terragrunt state mv "module.network.azurerm\_subnet.subnet[1]" "module.private-network.azurerm\_subnet.subnet[1]"

terragrunt state mv "module.network.azurerm\_subnet.subnet[2]" "module.private-network.azurerm\_subnet.subnet[2]"

terragrunt state mv module.network.azurerm\_virtual\_network.vnet module.private-network.azurerm\_virtual\_network.vnet

由於這個範例剛好 mv 一整個 module，我們也可以

terragrunt state mv module.network module.network-private

檢視 state list 中的最新狀況

terragrunt state list

module.private-network.data.azurerm\_resource\_group.network

module.private-network.azurerm\_subnet.subnet[0]

module.private-network.azurerm\_subnet.subnet[1]

module.private-network.azurerm\_subnet.subnet[2]

module.private-network.azurerm\_virtual\_network.vnet

**plan & review**

**how state mv angers your colleage**

從成功更改 state 這邊開始，會產生多人協作問題

* 本地的 .tf 已經 rename
* remote 的 state 已經 state mv
* 很有可能本地的 .tf 還沒有 merge 到 master，而是在另外一個 branch 上執行 state manipulation
  + 因為如果 master 有做 auto plan & auto apply 的話，merge 進去會直接執行 4 destroy, 4 create 的 plan，這不是我們想要的

在本地 .tf merge 進去之前，master branch 上 plan，會出現 4 destroy, 4 create，這是為何？

* state 已經變成本地 branch 的形狀了，也就是變成 module.private-network
* master 上面仍然是 module.network，這時他會找不到 state module.network，而發現多了 module.private-network，於是產生 4 destroy, 4 create

卡在這裡的風險，就是其他同事剛好也在 plan 的話，看到 plan 一定超困惑，不曉得發生什麼事情

如何用 terraform 惹怒同事：改 state 不講

所以 state manipulation PR 需要走特別的開發流程

* nofity all team: 通知全 team 是第一步
* 通知要更改的 root module path，別的成員不要來 plan / apply
* rename 的 branch 與 PR 先發，進行 code review
* review 完後，不走 PR merge + auto plan 與 auto apply，原因上面說過，會爆炸
* 直接使用 PR 中的 branch，進行 state manipulation
* state 變更完成後，執行 PR merge
* auto plan 與 auto apply 會正常通過，因為 state 已經 rename 成新的名稱了

**Homework**

* 再把 module.private-network rename 回 module.private-network XD，我們是為改而改，現在請把它改回來
  + 需要重新 init 嗎？
  + state list 與 plan 確定結果