**Review**

本篇講解進階的 terraform state 操作，請複習[基本State 操作](https://ithelp.ithome.com.tw/articles/03-basic-state.md)與[基本 Backend](https://ithelp.ithome.com.tw/articles/04-basic-backend.md) 的相關概念

* state 內含 terraform 運作產生的最終資料與中間產物（ex. 變數，random resource...等）
* state 內含連結 .tf resource 與公有雲上遠端 resource 的，terraform 仰賴 resource metadata （ex. 遠端 resource 的 id），來對應 .tf resource
* state 實作邏輯，由各家 provider 實作，state 內也會包含 provider 與 public cloud api 溝通時需要的資料

**Standard state workflow**

一般來說，我們可以完全仰賴 provider 自動管理 state，不需要手動干涉 state 內容。此時 state 的 workflow 很單純：針對每個 root module

* terraform init 初始化 state
* terraform plan
  + lock state
  + refresh state 更新 state 與遠端 resource 的狀態
  + provider 依據最新 state 與 .tf 的差異，計算應該變更計畫
* terraform apply
  + lock state
  + 進行 apply，由 provider 發出 api 到 public cloud
  + 等待 public cloud response，並依據 response 更新 state

前面 18 天的課程，我們都是依照上面的流程，讓 terraform 自動管理 state

**state manipulation: the bad**

更改 state 有其風險，State manipulation 有賺有賠（？），更改前應詳閱官方文件說明書

[Terraform 官方文件對於 state manipulation 的描述](https://www.terraform.io/docs/cli/state/index.html) 標記 important note

Important: Modifying state data outside a normal plan or apply can cause Terraform to lose track of managed resources, which might waste money, annoy your colleagues, or even compromise the security of your operations. Make sure to keep backups of your state data when modifying state out-of-band.

重要！在正常 plan / apply 以外的流程更改 state，可能造成 terraform 無法追蹤遠端 resource

* 可能會浪費雲端 resource 的花費
* 高機率會惹惱同事
* 或是造成安全性的漏洞  
  如果要操作 state manipulation，務必先備份

**State addressing**

在進到更改 state 之前，要來細看 [terraform 是如何 address state](https://www.terraform.io/docs/cli/state/resource-addressing.html)

以 azure/foundation/compute\_network 為例說明。首先回憶一下 .tf 內容

* 使用 local module modules/compute\_network
* module 內部又使用 https://github.com/Azure/terraform-azurerm-network 的 remote module
* 忘記的話記得去 <https://github.com/Azure/terraform-azurerm-network> 回憶一下
* remote state 在 azure blob storage 上，可以上 azure console -> storage browser 看到 .tfstate

由於我們已經 apply 過，這邊使用 state list 列出現有的 state

* state 的表示是用 state address 代表
  + resource 與 module block 的名稱可能重複，但是一個 path 底下的 block name 都是唯一的
  + 也就是說每個 resource address 都代表一個 resource / data source

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

我們這個 root module (azure/foundation/compute\_network) 產生五個 resource

* 一個 data source，意思是這個 data block 從遠端獲取一個 data source，例如遠端的物件，以及物件的參數，把 data 存到 state 中，讓其他 resource 引用。但本身不會 create 遠端的 resource
  + 詳見[後面可能有機會講到的 data source & resource](https://ithelp.ithome.com.tw/articles/10266597)
* 一個 multi-instance resource module.network.azurerm\_subnet.subnet(s)
  + 這個 resource 是 multi-instance，在呼叫 / address 其中一個 instance 時候，可以使用 module / resource index (ex. [0])
  + 如果 module / resource 是 collection，addressing 會加上 index
* 一個 single instance resource module.network.azurerm\_virtual\_network.vnet

module / resource 的命名也很直觀

* module + name
* resource type + name
* 然後可以一直串下去，越多層越長，複雜的 module 底下 address 接連到天邊...(遠目)

module.<module-name>.azurerm\_virtual\_network.<resouce-name>

**Why state manipulation**

上面把 state manipulation 說得這麼可怕，那 terraform 都自動把 state 維護好，我們有什麼動機需要手動來更改 state？

實務上比較常見到的例子

* inspect state，我只是想看一下 state 內容
* 使用 terraform taint 來強制 resource recreate
  + 如果 module 有使用 lifecycle meta-argument 中的 ignore changes，就有機會搭配使用
  + 或是 provider 的新功能，但支援不是那麼完整的時候，我們強迫 recreate
* resource / module 重新命名，rename resource 在 terraform 會被視為 delete + create
* import 原本不是 terraform 產生的 resource，導入 terraform 管理，import 到 terraform 中
* Disaster Recovery，因為出事了阿北，state 壞了只好來手動修

**Let's talk about resource / module rename**

rename resource / module 應該是常見的需求，特別是針對開發中的 module，會希望讓 resource 的名稱更直觀

* naming 也會隨 module 的編輯而逐漸改變
* 例如本來 module 中只有一個 vm 所以命名為 vm.main，之後覺得要做 High Availability，所以 rename 變成 vm.master + vm.slave[0] + vm.slave[1]

從 resource addressing

* module + name
* resource type + name

如果我們把 module.network.azurerm\_virtual\_network.vnet 的 azurerm\_virtual\_network.vnet rename 成為 azurerm\_virtual\_network.main

* address 改變了，terraform 有辦法辨識兩個 resource address 是相同 resource 嗎？
* 答案是不行，terraform 會看到
  + module.network.azurerm\_virtual\_network.vnet 不見了
  + 多一個 module.network.azurerm\_virtual\_network.main
  + 因此覺得使用者是想要
    - 刪掉有 state 存在，但是 .tf 中不見的 module.network.azurerm\_virtual\_network.vnet
    - 沒有 state 存在，但是 .tf 中出現的 module.network.azurerm\_virtual\_network.main 應該要產生

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

}

+ module.network.azurerm\_virtual\_network.main {

}

1 to create, 1 to delete

這也是 terraform，應該說是 RESTful API 常見的問題

* 如何從 api 端，判斷使用者的意圖，是希望 resource 的 rename，而不是 delete + create

在這種情形下，如果我們希望的是 rename，便可以更改 state

* 把 state 中的 module.network.azurerm\_virtual\_network.vnet -> module.network.azurerm\_virtual\_network.main
* 於是 terraform 就會看到 .tf 中有 module.network.azurerm\_virtual\_network.main，state 中也有 module.network.azurerm\_virtual\_network.main，兩個是對在一起的
* 而且 remote resource 的 id 仍然能對照公有雲上正確的 resurce

state mv 方法很多種，例如：

* 直接打開 editor，對準 terraform.tfstate 檔案，vim 下去直接把路徑改掉 (DANGER)
* 或是使用 terraform 指令， terraform state mv 來

**Don't do this! Edit Local State with Editor**

State manipulation 有其風險有賺有賠（？）工程師應詳閱官方文件公開說明書。

今天我們遇上上面描述的情形，被迫要手動調整 state，然而 state 更改方式許多種，使用 editor 直接 edit 大概是最容易出錯的一種

* 複雜的 nested json，看了眼睛痛，沒人想用手維護 json
* 組織越複雜的 json 越容易犯錯
* non-programmatic approach，富含 human error，難以復現 reproduce，難以自動化與標準化

當使用 local state 進行變更時（ex. apply），本地會有一個版本的 terraform.tfstate.backup 檔案，是 terraform 使用 local state 的時候，一個貼心的小功能。如果

如果 terraform.tfstate 跟 terraform.tfstate.backup 都改壞了，那復原的步驟就會變得非常麻煩，而且一個不小心就會弄壞遠端的 resource。這在 prod 發生的話不是開玩笑的

附帶一提，這時就體現出 state versioning 的優勢，如果使用的 backend 有支援 state versioning，遠端的 backend 上會保留非常多以前 apply 的 state，萬一真的不幸搞壞最近幾個版本，還是可以 checkout 更早版本的 state，協助回朔

然而，版本離越遠，回朔的過程越痛苦。這是一個惹惱同事的好方法 ;)

**Homework**

爛方法沒試過不知道他有多爛

* 使用 local state apply，或是使用之前 \_poc 內使用 local backend 的範例
* rename 過去寫過的任何 remote
* 備份 terraform.tfstate
* 只用最愛的 editor 直接編輯 terraform.tfstate，更改 state resource address 到新的 resource.name
* 再次 terraform plan，看 terraform 是正常運作