上一講 [Day 02-是在 Hello？什麼都要 Hello 一下之 Hello Terraform](https://ithelp.ithome.com.tw/articles/10259260)，我們操作 terraform 指令，來 create / update / destroy 遠端的資源。在執行完成 terraform apply 後，本地資料夾會產生一個 terraform.tfstate 檔案。

**初探 .tfstate**

首先，我們看一下 terraform.tfstate 檔案的內容。你可以使用文字編輯器，或是透過 shell 與 [jq](https://stedolan.github.io/jq/" \t "_blank) 工具來檢視 State

cat terraform.tfstate

cat terraform.tfstate | jq keys

[

"lineage",

"outputs",

"resources",

"serial",

"terraform\_version",

"version"

]

其中 .resources 是紀錄 .tf 檔案產生的資源，在遠端的 instance 的實際資料。換句話說，Terraform apply 後產生資源滿足 .tf 的描述，而實際在遠端的實體是哪一個，有哪些資料，紀錄在 State 中。

再舉個例，如果我們想要產生多個不同的 \_poc/foundation/\*.tf 中的 resource，複製 .tf 檔案在 apply，會獲得另一組 foundation resource，如另一組 resource group 與 storage account，有相同的參數，但有不同的遠端 id。

我們還可以比較 .tfstate 的內容，與 azure console 上看到的內容，更能理解兩者個關係。

**State 設計**

根據[官方文件](https://www.terraform.io/docs/language/state/purpose.html) 描述 State 的設計與功能，這邊簡述幾個重點，底下會有範例詳述

* Mapping to the Real World：只有 .tf 檔案的話，並不能明確的對應到遠端的 resource，Terraform 託管需要的 metadata 並存放在 State 中
* Metadata：除了 mapping 必須資料外，State 也而外存放 Terraform runtime 需要的資料（之後章節描述）
* Performance：Terraform plan 與 apply 需要先 refresh 遠端的狀態，State 作為一層資料 Cache，可以加速工作流程
* Syncing：多人協作

**State 與 Backend**

釐清名詞 State

* State 是一個 terraform 的核心架構，在原始碼中是一個抽象層，意思是 terraform 支援許多不同的 State 實作，不同實作的機制不同，但都能滿足上面描述的 State 概念，滿足 Terraform 執行時的需求
* 本地的 .tfstate 是使用 [Local Backend](https://www.terraform.io/docs/language/settings/backends/local.html) 這個 Backend 實作來管理 State 時，於本地儲存的 State 檔案。換句話說，如果不使用 Local Backend，本地就不會產生這個 .tfstate

**Issues**

使用 Local State 有許多好處

* 不需設定就可直接使用
* 本地 .tfstate 檔案可以快速檢視，輕易的編輯 State
* 注意：編輯 State 造成損壞可能會造成 Terraform 執行錯誤，請見 [Day 20-state inspection-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之二](https://ithelp.ithome.com.tw/articles/10268643)

然而使用 Local State 也有以下幾個問題

**沒有 .tfstate 檔案就無法使用 Terraform**

Q: 沒有 .tfstate 檔案就無法使用 Terraform 嗎？可是我的 .tf 檔案裡 name 寫得很清楚，這樣 Terraform 抓不到遠端相同名字的資源嗎？

我們可以做個實驗，切換到另外一個 foundation cloned 資料夾，內容與 foundation 完全一致。

cd ../foundation\_cloned

terraform init && terraform plan

plan 的結果是什麼？

是否令人有些疑惑？

* 內容完全一樣的 .tf 檔案，plan 出來的結果卻是要重新產生所有資源
* 實際執行 terraform apply，provider 會回覆錯誤 'A resource with the ID ".../resourceGroups/terraform-30-days-poc" already exists"'

我們原先預想透過相同的 .tf 檔案來管理 resourceGroup/terraform-30-days-poc，然而 Terraform 無法追蹤遠端已經存在的 resource group，而是選擇 create 一個新的 resource group，provider API 送出後，遠端的 Azure 回傳錯誤

更明確的說明：沒有 State 便無法 mapping .tf resource，與遠端 resource

我們可以進一步檢視，.tfstate 檔案的哪些內容，使得一邊可以正常使用 Terraform，一邊何謂產生錯誤。使用 [jq](https://stedolan.github.io/jq/" \t "_blank) 工具來檢視 State 中與 resource gropu 有關的資料。

cat terraform.tfstate | jq '.resources[0]'

{

"mode": "managed",

"type": "azurerm\_resource\_group",

"name": "rg",

"provider": "provider[\"registry.terraform.io/hashicorp/azurerm\"]",

"instances": [

{

"schema\_version": 0,

"attributes": {

"id": "/subscriptions/6fce7237-7e8e-4053-8e7d-ecf8a7c392ce/resourceGroups/terraform-30-days-poc",

"location": "southeastasia",

"name": "terraform-30-days-poc",

"tags": null,

"timeouts": null

},

"sensitive\_attributes": [],

"private": "eyJlMmJmYjczMC1lY2FhLTExZTYtOGY4OC0zNDM2M2JjN2M0YzAiOnsiY3JlYXRlIjo1NDAwMDAwMDAwMDAwLCJkZWxldGUiOjU0MDAwMDAwMDAwMDAsInJlYWQiOjMwMDAwMDAwMDAwMCwidXBkYXRlIjo1NDAwMDAwMDAwMDAwfX0="

}

]

}

cat terraform.tfstate | jq '.resources[0].instances[0].attributes'

{

"id": "/subscriptions/6fce7237-7e8e-4053-8e7d-ecf8a7c392ce/resourceGroups/terraform-30-days-poc",

"location": "southeastasia",

"name": "terraform-30-days-poc",

"tags": {},

"timeouts": null

}

Terraform 產生 resource group 後，將 azure API 回覆的 resources 各項參數紀錄在 .tfstate 中。下次要再進行編輯時，我們編輯 .tf 檔案，Terraform 則會依據 .tfstate 檔案去 mapping 遠端的 resource group，進行 plan 與 apply。

熟悉 RESTful API 的朋友可以這樣想：使用 POST API 產生物件時，後端 server 會回傳 id 在 response body 中，下次要編輯這個相同物件，則需要使用 id 作為辨識。Terraform 其實是相同的原理，只是在這個例子中，Terraform 協助託管了 id 這個 metadata。

**協作問題**

既然使用 terraform 時，必須仰賴 State .tf 檔案，那是否一起協作的團隊成員就必須要取得 .tfstate 檔案，才能正確的操作 Terraform？

是的，這也是使用 Local Backend 的 .tfstate 檔案，最大的問題，會造成多人協作十分困難。過往舊版的 Terraform 有幾個妥協的做法：

將 .tfstate 檔案與 .tf 檔案一起納入版本控制系統，具體流程可能是這樣

* 編輯完 .tf 檔案後，commit
* 依據新的 .tf 檔案，執行 terraform plan / apply，產生新的 .tfstate 檔案後，commit .tfstate 檔案
* 推上版本控制系統 (ex. Github) 其他團隊成員只要 git pull 最新的 .tf 與 .tfstate 檔案，就可以正確使用 Terraform

理想上是這樣，但實務上還是非常不便

* 必須要確定 state 以一直最新的，團隊成員如果有沒 commit 的 state，馬上會造成 conflicts 與開發 blocking
* .tf 有事需要 review 與測試
* State 中可能包含 sensitive 資料，sensitive 資料不宜放到版本控制系統

本課程不建議把 .tfstate 加入到版本控制，本 repository 已經將.tfstate 加入到 .gitignore 中。

**sensitive 資料與安全性**

所有在 terraform plan / apply 中產生的 data 與 meta-data，都會紀錄在 tfstate 中，那是否有一些 State 內容是敏感資料，不希望讓他人看到？

我們可以使用 \_poc/user/ 為例子。首先檢視一下內容 .tf 檔案，這個檔案使用 Terraform random password 來產生一組隨機密碼，然後使用這組密碼作為新的 User 的登入密碼。

*# \_poc/user/ad\_user.tf*

resource "random\_password" "terraform" {

length = 16

special = true

override\_special = "\_%@"

}

resource "azuread\_user" "terraform" {

user\_principal\_name = "terraform@chechia.net" *# Need valified domain on Azure AD*

display\_name = "Terraform Runner"

mail\_nickname = "terraform"

password = random\_password.terraform.result

}

產生 user terraform 資源

terraform apply

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

Changes to Outputs:

+ password = (sensitive value)

+ user\_principal\_name = "terraform@chechia.net"

...

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

Outputs:

password = <sensitive>

user\_principal\_name = "terraform@chechia.net"

Apply 完成後，我們可以看一下 \_poc/user 產生的資料，這邊有點有趣

* 我們 .tf 檔案中有 output.tf，將產生的資料輸出，讓外部可以使用。例如，使用者可取得這組密碼來進行登入
* 然而 terraform output 卻輸出 password =

我們可以在用其他方法印出 sensitive 資料：

terraform output

password = <sensitive>

user\_principal\_name = "terraform@chechia.net"

terraform output -json

{

"password": {

"sensitive": true,

"type": "string",

"value": "QfgdxMwVRKr41nHG"

},

"user\_principal\_name": {

"sensitive": false,

"type": "string",

"value": "terraform@chechia.net"

}

}

cat terraform.tfstate | jq '.resources[1].instances[0]'

{

"schema\_version": 0,

"attributes": {

"id": "none",

"keepers": null,

"length": 16,

"lower": true,

"min\_lower": 0,

"min\_numeric": 0,

"min\_special": 0,

"min\_upper": 0,

"number": true,

"override\_special": "\_%@",

"result": "QfgdxMwVRKr41nHG",

"special": true,

"upper": true

},

"sensitive\_attributes": [],

"private": "bnVsbA=="

}

我們可以看到，所有密碼還是明碼的印出來。

* 就算使用 sensitive 關鍵字，本地產出的資料，Terraform 還是需要透過 State 管理
* Local Backend 的 .tfstate 是沒有任何加密的，取得檔案就取得密碼，取得 State 就可以取得隱私資料
* 如果使用版本控制，這個密碼也會存在 Git objejct 中

更好的方式是

* 使用其他 Backend 來管理 State
* 使用有加密功能的 Backend

下堂，我們要來使用遠端的 Backend 與 State

**Homework**

嘗試使用 terraform state 的幾個指令

terraform state --help

terraform state list

terraform state show

terraform destroy 刪除所有本課程的產物

閱讀以下官方文件

* <https://www.terraform.io/docs/language/settings/backends/index.html>
* <https://www.terraform.io/docs/language/settings/backends/local.html>
* <https://www.terraform.io/docs/language/state/purpose.html>

**References**

* <https://www.terraform.io/docs/language/state/index.html>
* <https://www.terraform.io/docs/language/state/sensitive-data.html>