本篇介紹 terraform 基本設定與操作指令，並使用 terraform 管理公有雲上的 resource

[這裡指的 resources 有明確定義，底下會說明](https://ithelp.ithome.com.tw/articles/10259260)

**Prerequisite**

這個課程包含許多公有雲上的 resource 範例，建議參與 workshop 的觀眾都要有 Azrue 的 account，[取得Azrue account 設定方法在這邊](https://ithelp.ithome.com.tw/articles/01-get-started.md" \t "_blank)

如果是第一次使用公有雲，或是想省一點錢的朋友，可以參考 [Azure 公有雲提供的 Free Tier 免費額度](https://azure.microsoft.com/zh-tw/free/free-account-faq?WT.mc_id=AZ-MVP-5003985)

**Hello Azure**

本課程主要以 Azure Cloud 為主要範例，然而不管使用哪做公有雲，Terraform 操作的基本觀念都類似

[依照官方文件安裝 Azure-cli](https://docs.microsoft.com/zh-tw/cli/azure/install-azure-cli?WT.mc_id=AZ-MVP-5003985)

brew install azure-cli

az version

然後使用 az login，透過網頁登入帳號，這個指令會在本機留下 azure 的 credential

az login

[{

"cloudName": "AzureCloud",

"homeTenantId": "1234567-my-home-tenant-id",

"id": "1234567-my-id",

"isDefault": true,

"managedByTenants": [],

"name": "my-subscription",

"state": "Enabled",

"tenantId": "1234567-my-tenant-id",

"user": {

"name": "my-email",

"type": "my-user"

}

}]

...

**Auth to Azure**

使用 terraform 管理 azure cloud 的 resources，自然 terraform 需要取得有效的 auth credentials，才能存取 Azure Cloud 的 API

* 由於我們已經執行 az login，az-cli 會在預設的路徑留下 credential (~/.azure/)
* terraform azure provider 會搜尋預設的路徑，使用 azure credential 管理遠端公有雲上的 resources
* 可以看一下 az-cli 在本機留下什麼 登入資訊與 credentials

ls -al ~/.azure/

cat ~/.azure/azureProfile.json

**Terraform content**

在執行任何 terraform 程式碼前 .tf 或 .hcl，我們先看一下內容

以 azure/\_poc/foundation/ 為例

cd azure/\_poc/foundation/

ls

main.tf

provider.tf

output.tf

* provider.tf 描述了 [terraform provider](https://ithelp.ithome.com.tw/articles/10259260) 的設定，terraform 本身是抽象層的管理工具，實際上 [resource 的行為，會依賴各家廠商的 provider 實作，一樣我們底下會細講](https://ithelp.ithome.com.tw/articles/10259260)
* main.tf 是主要的 resources，內容包含
  + [random id](https://registry.terraform.io/providers/hashicorp/random/latest/docs/resources/id) \* 1
  + [azurerm resource group](https://docs.microsoft.com/zh-tw/azure/azure-resource-manager/management/overview#resource-groups?WT.mc_id=AZ-MVP-5003985) \* 1
  + [azurerm storage account](https://docs.microsoft.com/zh-tw/azure/storage/common/storage-account-overview?WT.mc_id=AZ-MVP-5003985) \* 1
  + [azurerm storage container](https://docs.microsoft.com/zh-tw/azure/storage/blobs/storage-blobs-introduction?WT.mc_id=AZ-MVP-5003985) \* 1
* output.tf 定義參數輸出，有點像高階程式語言的 function return value，在這個 [terraform module (後面細講)](https://ithelp.ithome.com.tw/articles/10259260) 產生的參數可以透過 output 輸出

三個 azure 的 resource，如果不熟的就需要查一下上面的官方文件。這邊提供不精準的對照

* azure / aws resource group ~= gcp project
* azure storage container ~= aws s3 / gcp storage

上面是後續 terraform 會需要的 azure resource，這邊嘗試把他產生出來

**First Terraform Command: terraform init**

終於，執行第一個 terraform command，terraform init

terraform init

Initializing the backend...

Initializing provider plugins...

- Finding hashicorp/azurerm versions matching "~> 2.65"...

- Installing hashicorp/azurerm v2.65.0...

- Installed hashicorp/azurerm v2.65.0 (signed by HashiCorp)

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!

所有 terraform module 要正確運作前都需要 init，init 做了許多事情，這邊先看 log

* init backend，[第三堂 basic backend 會細講](https://ithelp.ithome.com.tw/articles/03-basic-backend.md" \t "_blank)
* 下載 provider，總之 terraform 核心只是抽象，具體 resource 與 resource 的行為，要由 provider 決定。[provider 在第 ? 堂 會細講](https://ithelp.ithome.com.tw/articles/10259260" \t "_blank)

**Plan**

[參考文件的說明 terraform plan](https://www.terraform.io/docs/cli/commands/plan.html)

簡單說就是會 diff .tf 檔案，以及 azure cloud 上面的實際內容，有落差會顯示差異，然後輸出變更計畫。[複雜的說我們大概第四天見 basic state](https://ithelp.ithome.com.tw/articles/10259260" \t "_blank)

terraform plan

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:

*# azurerm\_resource\_group.rg will be created*

+ resource "azurerm\_resource\_group" "rg" {

+ id = (known after apply)

+ location = "southeastasia"

+ name = "terraform-30-days"

}

*# azurerm\_storage\_container.main will be created*

+ resource "azurerm\_storage\_container" "main" {

+ container\_access\_type = "private"

+ has\_immutability\_policy = (known after apply)

+ has\_legal\_hold = (known after apply)

+ id = (known after apply)

+ metadata = (known after apply)

+ name = "tfstate"

+ resource\_manager\_id = (known after apply)

+ storage\_account\_name = "tfstate"

}

...

Plan: 3 to add, 0 to change, 0 to destroy.

────────────────────────────────────────────────────────────────────────────────────────────────────

main.tf 裏面寫了 4 個 resources，一個是 random id，另外三個是 azure resource。這邊 terraform plan 的計畫，認為應該要 create 三個物件。

Plan: 3 to add, 0 to change, 0 to destroy.

依據 .tf 與實際公有雲上的 resource 差異，也有可能會有 change 或是 destroy

目前就先當作是，工程師寫下想要的 .tf 狀態，terraform 會幫你『多退少補』

**Apply**

Plan 之後，我們知道 terrform 計畫的結果，如果與我們想得符合（ex. 3 to add plan 符合我們的預期），表示 .tf 與 plan 沒有問題，這時就可以使用 terraform apply 真的把 request 送到 Azure 上，產生我們缺少的 resource

terraform apply

Do you want to perform these actions?

Terraform will perform the actions described above.

Only 'yes' will be accepted to approve.

Enter a value: yes

重點：養成好習慣，一定要把 apply 的內容仔細看完，再輸入 yes。不看直接 apply 會遭到[第一天引言所說的報應](https://ithelp.ithome.com.tw/articles/01-introduction.md" \t "_blank)

NOTE: Always double check before type yes.

**Terraform local files**

apply 下去，會需要一些時間，terraform 會回報已經花費的時間。

執行完成後，可以觀察到兩件事

* 透過 azure web console 可以看到真的有 resource group, storage account, ... 物件產生出來
  + 這是我們想要完成的工作：透過 terraform 管理遠端的 azure 物件，成功！
* 本地的 azure/\_poc/foundation/ 資料夾下多了一些東西

這邊看一下 terraform 在本地留下什麼東西

*ls* -*al*

*drwxr*-*xr*-*x* .*terraform*

-*rw*-*r*--*r*-- .*terraform*.*lock*.*hcl*

-*rw*-*r*--*r*-- *main*.*tf*

-*rw*-*r*--*r*-- *output*.*tf*

-*rw*-*r*--*r*-- *terraform*.*tfstate*

-*rw*-*r*--*r*-- *terraform*.*tfstate*.*backup*

這些檔案對 terraform 都有不同功能，我們預設會把這些檔案隱藏，並且 gitignore。在現階段我們只要知道 terraform apply 完後在本地產生一些東西。

* terraform.tfstate: 是 Terraform state 檔案 [大約在第三天 basic backend](https://ithelp.ithome.com.tw/articles/03-basic-backend.md) 與 [第四天 basic state](https://ithelp.ithome.com.tw/articles/04-basic-state.md) 會細講這些檔案的用途

**plan again**

既然 apply 完成，我們可以在 terraform plan 一次看看，會產生出什麼計畫

terrafrom plan

...

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

這表示目前的 azure/\_poc/foundation/ 資料轄下的 .tf 內容，與遠端的 azure 上的 resource 是相同的

* 遠端的狀態是我們想要控制達成的狀態
* 見 code 如見人，不用上雲看

**Potential Error: name is already taken**

這邊有個常見錯誤先提醒：azrue storage account 的名稱是 global unique，(~所以要搶 id~)，命名有衝突的話， terraform API 發到 Azure 上，Azure 會回傳 error

╷

│ Error: Error creating Azure Storage Account "tfstate": storage.AccountsClient#Create: Failure sending request: StatusCode=0 *-- Original Error: autorest/azure: Service returned an error. Status=<nil> Code="StorageAccountAlreadyTaken" Message="The storage account named tfstate is already taken."*

│

│ with azurerm\_storage\_account.main,

│ on storage\_account.tf line 1, in resource "azurerm\_storage\_account" "main":

│ 1: resource "azurerm\_storage\_account" "main" {

│

╵

請使用低碰撞機率的命名

細節請 google terraform azure storage account

[Hashicorp doc: storage account](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/storage_account)，的參數有描述說明

name - (Required) Specifies the name of the storage account. Changing this forces a new resource to be created. This must be unique across the entire Azure service, not just within the resource group.

**3 steps Terraform**

terraform 三步驟

* init
* plan
* apply

基本 debug 三步驟

* 讀 azure cloud 文件
* 讀 hashicorp resource 文件
* retry

養成查文件的習慣，不只是 terraform 需要，想要熟悉 public cloud infrastructure 都要時常查找文件

**About random**

resource random id 後續課程有機會再聊

**Remove Storage Container**

接下來嘗試 destroy 所有剛剛產生的 resource

* 使用最愛的編輯器，更改
* 在所有 resource block {} 前面在加 comment，或是直接刪除

vim azure/foundation/main.tf

...

#resource "azurerm\_storage\_container" "main" {

# name = "tfstate"

# storage\_account\_name = azurerm\_storage\_account.main.name

# container\_access\_type = "private"

#}

...

然後我們再次 plan 這個 directory 的 module

terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are

indicated with the following symbols:

- destroy

Terraform will perform the following actions:

*# azurerm\_storage\_container.main will be destroyed*

- resource "azurerm\_storage\_container" "main" {

- container\_access\_type = "private" -> null

- has\_immutability\_policy = false -> null

- has\_legal\_hold = false -> null

- id = "https://tfstatef4380b8b1152083e.blob.core.windows.net/tfstate" -> null

- metadata = {} -> null

- name = "tfstate" -> null

- resource\_manager\_id = "/subscriptions/6fce7237-7e8e-4053-8e7d-ecf8a7c392ce/resourceGroups/terraform-30-days/providers/Microsoft.Storage/storageAccounts/tfstatef4380b8b1152083e/blobServices/default/containers/tfstate" -> null

- storage\_account\_name = "tfstatef4380b8b1152083e" -> null

}

Plan: 0 to add, 0 to change, 3 to destroy.

terraform 的計畫顯示想要 destroy 3 個物件

Plan: 0 to add, 0 to change, 3 to destroy.

這邊注意：

* 我們確實是想要刪除遠端 resource，所以把 main.tf 裡面的 resource block {} 刪除
* 確定是我們要的結果，並且仔細檢查 destroy 掉的 resource 內容
* 才執行 terraform apply

destroy 的時候永遠 double check，很多infrastructure destroy 掉，就是『趴！沒了！』，例如[前言所提到的 database](https://ithelp.ithome.com.tw/articles/01-introduction.md" \t "_blank)

* 刪掉 infrastructure 有可能會影響其他 infrastructure，如果彼此有依賴關係的話
  + ex. 刪除 storage account 同時也會讓 storage container 失效，要格外注意
  + 正確使用 terraform 可能可以幫你預測這些依賴性的效果

確認後就放心 apply

terraform apply

Do you want to perform these actions?

Terraform will perform the actions described above.

Only 'yes' will be accepted to approve.

Enter a value: yes

一樣需要花飛一點時間，讓 Azure destroy resource

* 完成後去 azure web console 檢查，3 個 azure resouces 都刪除了

**Resources**

這邊提到的 [resources 在官方文件中也有明確的定義](https://www.terraform.io/docs/language/resources/index.html)，有幾個不同面向

* 一個是本地 .tf 中的 resource 描述，是 hcl 語言。[細節大約是在第 14 天左右會說明](https://ithelp.ithome.com.tw/articles/13-syntax.md" \t "_blank)
* 先當作一個一個 resource block {}
* 一個是遠端實際的 azure resource / 或說是 azure object，例如一個 resource group / 一個 VM / 一個 VM

resource 的行為由 provider 定義，azurerm provider 就定義：如何把 .tf resource 變成 azure resource 的 API request，丟到 Azure 上去，產生 resource

**Directory Structure**

看一下資料夾結構

tree -L 1 azure

azure

├── KNOWN\_ISSUES.md

├── \_poc

├── dev

├── env.tfvars

├── foundation

├── modules

├── prod

├── stag

├── terragrunt.hcl

└── test

7 directories, 3 files

terraform plan 預設會以一個 directory 作為 root module，

* 使用高階程式語言比喻，可能是程式進入點的概念

我們剛開始會使用 \_poc 的內容做示範，只是為了展示 terraform 功能，之後會把 \_poc 內的 resource 全部移除

底下這些則是依照不同環境切分的資料夾，是 [terragrun 官方建議的資料夾結構，我們在第 7-8 天左右會細講](https://ithelp.ithome.com.tw/articles/07-terragrunt.md" \t "_blank)

* foundation
* dev
* stag
* prod

**Homework**

Now, try the following practice

1. 增加一個全新的 storage container resource block，也就是說 apply 玩會有兩個 storage container
2. 另外找一個 azure/poc 內的資料夾來 init, plan, apply 看看
3. 嘗試更改 resource block {} 裏面的參數 variables，例如 name, ...等等設定
4. 檢查 local state 檔案

* 使用最愛的編輯器，打開 terraform.tfstate 檔案看看內容
* 執行 terraform destroy
* destroy 後，再次檢查 terraform.tfstate 的內容

**Summary**

第二天，我們跟 terraform 説 hello。目前為止我們知道

* terraform 以某種機制 sync
  + 本地 .tf 內容的描述
  + 遠端 azure cloud 的實際 resource
* terraform 會在 plan 結果顯示
  + add, change, 或 destroy
  + 多退少補
* Terraform 宣告式的 declarative 描述 resource block
* provider 會處理實際 sync 的這個行為
* 養成好習慣：每次 apply 都 double check