Backend 是初學 Terraform 的核心概念，本章節會簡述 Backend，並介紹 Remote Backend 的原理。

[上一講 Day 03-Terraform State 之你的 Local State 不是我的 State](https://ithelp.ithome.com.tw/articles/10259586) 我們討論 State 基本觀念，了解 Terraform 是如何使用 State ，協助與遠端 resource 的管理。

上堂課最後，我們也探討使用 Local Backend，將 State 以 .tfstate 檔案存放本機，各種優點與缺點。這裡我們稍微回憶一下，Local Backend 的問題：

* Local Backend 不支援多人協作，其他人無法存取本機的 .tfstate 檔案，就無法正確操作 Terraform
* State 內含敏感資料，.tfstate 檔案並未加密，產生的資料是明碼存放。也不宜上傳 git 等版本管理工具
* State 內含 Terraform lock，可以避免多人同時對相同 resource 一起 apply 變更，Local Backend 不支援多人協作，因舞也無法使用 lock 保護遠端的 resource

這些問題該如何解決？是否有更適合多人協作的 Backend？

**Terraform Backend**

根據[官方文件](https://www.terraform.io/docs/language/state/backends.html" \t "_blank)，Backend 主要的功能，就是

* 儲存 State 的資料，提供團隊多人協作
* 提供 State locking，避免多人同時修改

Terraform 整合[許多 Backend](https://www.terraform.io/docs/language/settings/backends/index.html" \t "_blank) 供使用這選擇，這邊簡單介紹：

* Terraform
  + local: 前兩堂課使用的 local backend
  + remote: 官方提供，為 Terraform 訂做的 [terraform cloud](https://ithelp.ithome.com.tw/articles/app.terraform.io) 提供遠端 State  
    公有雲
  + azurerm: 使用 Azure 的 Azure Storage 與 Blob 儲存 State
  + gcs: 使用 Google Cloud Storage (GCS)
  + s3: 使用 AWS s3 儲存 State
  + cos: 使用 Tencent Cloud Object Storage 儲存 State
  + oss: 使用 Alibaba Cloud OSS
* 其他第三方
  + artifactory: jfrog artifactory 上儲存 State
  + swift: 使用 OpenStack Object Store
  + kubernetes: 使用 Kubernetes secret 儲存
  + pg: 使用 Postgres database 儲存
  + consul: 使用 hashicorp consul 儲存 State
  + etcd, etcdv3: 使用 etcd 作為 State 儲存

所有公有雲提供的 Backend 實作原理近似，挑選熟悉的平台參照課程的步驟即可。至於如何選擇適合的 Backend，互相比較與優缺點，稍後的課程再跟各位介紹。

本課程著重於公有雲，使用 azurerm 作為範例。

**Basic Azurerm Backend**

這邊講解 Azurerm Backend 的基本觀念，事實上非常單純

基本的要求，是使用 azure storage 存放 .tfstate 檔案，將放在本地的 .tfstate 放到 azure storage 上，團隊成員只要使用 terraform 執行相同的 .tf 檔案，Terraform 就可以自動取得遠端 azure storage 中的 .tfstate，進行使用。

這邊直接帶大家實際使用

**azurerm Backend configuration**

設定，兩個官方各提供一份說明文件

* [Azure Doc: terraform get-started](https://docs.microsoft.com/zh-tw/azure/developer/terraform/get-started-cloud-shell?WT.mc_id=AZ-MVP-5003985)
* [Terraform Backends: azurerm](https://www.terraform.io/docs/language/settings/backends/azurerm.html)

使用 azurerm storage 的前提

* azure account & credential(az login)
* azure subscription
* azure storage account
* azure storage container

前兩者在第一堂課時我們就是先準備好，已經使用了。後面兩個看起來十分眼熟？

沒錯，在前幾堂課程的範例 resource，便已經帶大家使用 Terraform 在 azure 產生這些 backend 所需的 resource。如果還沒產生的同學，可以到這裡來：

azure/\_poc/foundation

需要的 resource 都在這裡產生。使用 terraform output 便可以取得遠端 resource 的資料參數。

cd \_poc/foundation

terraform output

resource\_group\_name = "terraform-30-days-poc"

storage\_account\_name = "tfstate8b8bff248c5c60c0"

storage\_container\_name = "tfstate"

**terraform with remote backend**

之後新建的 .tf 資料夾，希望改用 azurerm backend，以 \_poc/container\_registry 為範例，使用 remote backend，在 provider.tf 的 terraform block 中，增加 backend {} block，依據 [backend 文件](https://ithelp.ithome.com.tw/articles/10259896) 說明，填入在 \_poc/foundation 中纖產生的參數。

terrform {

backend "azurerm" {

resource\_group\_name = "terraform-30-days"

storage\_account\_name = "tfstatee903f2ef317fb0b4"

container\_name = "tfstate"

key = "container\_registry.tfstate"

}

...

}

\_poc/container\_registry 內的 terraform {} 有設定使用 backend，terraform init 的時候便會使用

cd \_poc/container\_registry

terraform init

Initializing the backend...

Successfully configured the backend "azurerm"! Terraform will automatically

use this backend unless the backend configuration changes.

Initializing provider plugins...

- Finding hashicorp/azurerm versions matching "~> 2.65.0"...

- Installing hashicorp/azurerm v2.65.0...

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

Terraform has been successfully initialized!

以下的 plan 與 apply 的步驟都相同

terraform plan

terraform apply

在成功 apply 後，我們可以檢查本地的檔案，發現不再像之前有 .tfstate 檔案產生了

ls -al

.terraform

.terraform.lock.hcl

provider.tf

registry.tf

variables.tf

**parallel collaboration**

我們參照上節課的範例，使用 \_poc\_container\_registry\_cloned 來模擬一下多人協作時，State 是如何運作的。

首先，\_poc/container\_registry\_cloned 裡面只有 soft link 檔案。

cd \_poc/container\_registry\_cloned

ls -al

provider.tf -> ../container\_registry/provider.tf

registry.tf -> ../container\_registry/registry.tf

variables.tf -> ../container\_registry/variables.tf

init 後，本地產生 .terraform 資料夾，裡頭是 backend 的設定，以及最新從遠端下載的 terraform.tfstate，內容是完整的 .tfstate 檔案。

terraform init

Initializing the backend...

Successfully configured the backend "azurerm"! Terraform will automatically

use this backend unless the backend configuration changes.

Terraform has been successfully initialized!

ls -al

.terraform

provider.tf -> ../container\_registry/provider.tf

registry.tf -> ../container\_registry/registry.tf

variables.tf -> ../container\_registry/variables.tf

cat .terraform/terraform.tfstate

plan 與 apply 時，terraform 會自動將遠端的 State 拉到本地快取。

terraform plan

No changes. Your infrastructure matches the configuration.

如此便可以在不同資料夾，甚至多台電腦上，只要透過 terraform.backend{} 的設定，找到遠端的 State，就可以使用同一份 State 同時協作。

**State locking**

Azurerm backend 還有帶 [state locking](https://www.terraform.io/docs/language/state/locking.html) 功能。每當有團隊成員正在使用遠端 state 的時候，terraform 會自動在 azurerm storage state 上面打上 lock，當另一位成員試圖存取同一份 state 的時候，後來的成員的 terraform 指令會跳出 locked 訊息，並阻止 terraform 運作。

為何需要 state locking？

想像今天遠端有一台 VM instance，團隊成員 che 使用 terraform 修改 VM instance 的名稱，同時另一位成員也恰好想更改同一台 VM，如果沒有 state locking，兩個更改 VM 的 API requests 都一起送到 azure 上，會發生什麼事情？

這就會產生無法預期的錯誤，要看網路速度，以及 azure 對多重 request 的處理了，可能是先來先贏，或是後來的 overwrite，或是 request 衝突導致 azure 回傳錯誤；許多 request 修改 VM 需要時間，重複修改也可能會被公有雲直接拒絕，...，這些情形都很有可能損壞遠端的 resource。為了避免 apply conflict 發生，Terraform 使用 State locking

* 成員 A 開始 terraform apply
* Terraform 將遠端 State 打上 lock uuid
* 成員 A terraform apply VM update 費時兩分鐘，apply...
* 成員 B 剛好也上來改 VM，terraform plan
* 成員 B Terraform 試圖存取 state 時，就會看到 State lock，知道不是當前本機的 lock，跳出警告，並中斷 terraform 操作

Acquiring state lock. This may take a few moments...

Error locking state: Error acquiring the state lock: ConditionalCheckFailedException: The conditional request failed

Lock Info:

ID: 86ea49f6-255d-074e-97d8-dcd0e8d6c250

Path: \_poc/container\_registry/terraform.tfstate

Operation: apply

Who: chechia@chechias-MacBook

Version: 1.0.1

Created: 2021-07-29 13:13:27.894319 +0000 UTC

Info:

Terraform acquires a state lock to protect the state from being written

by multiple users at the same time. Please resolve the issue above and try

again. For most commands, you can disable locking with the "-lock=false"

flag, but this is not recommended.

* 成員 B 知道有其他成員在操作，摸摸鼻子去喝咖啡
* 成員 A 的 terraform apply 順利完成，遠端 VM 的狀態更新成 A 期待的結果，terraform 確認動作完成，自動解鎖 state lock
* 成員 B 再次 terraform plan，refresh state 時就會發現遠端 VM 已經變成 A 的形狀
* 成員 B 發現 state 有變，也知道其他成員改過 .tf 檔案，執行 git pull A 的程式碼，確保 .tf 檔案是最新的
* 成員 B 依據遠端狀態變更，在 .tf 檔案做作相應調整
* 成員 B 再次 terraform apply，完成工作

State locking 自動確保多人協作的工作流程是安全的，不會有 apply conflicts 發生。使用 terrform 宜盡量量使用支援 locking 的 backend。

**Backend additional features**

許多 backend 還有提供額外的功能，例如公有雲的 storage，不管是 azure storage, aws s3, 或是 google storage 都有自己的附加功能，使用這些附加功能也能加強 terraform state 的管理。

例如我們可以進一步設定 azure storage

* iam 存取控管，保護 state 資料
* 搭配 azure vault，進行 state 自動加解密，避免明碼檔案儲存
* azure blob 也有檔案版本控制功能，可以對不同版本的 state 進行管理，ex. 保存 20 個版本的 state

公有雲提供的 storage 是最容使用且管理方便，其他 backend 各有各自特性，選擇 backend 時宜多方比較。我們有空再談 backend 的比較。

**優劣**

Azurerm vs local

* 更方邊的多人協作
* state locking 保護
* 費用: azurerm 是計價服務，依據檔案容量與存旅遊量計費，由於 .tfstate 檔案並不會太大，使用公有雲儲存體是非常便宜
* 其他加值功能，這部分我們之後有機會再做介紹

**Source code**

附上兩個 Github Terraform 源碼，由於 Backend 與 State 的功能並不複雜，裡頭也沒有太過難的程式碼，熟悉 golang 的朋友不妨快速看過，會對 Terraform 有更明確的瞭解。細節若稍後有篇幅我們再來細講。

* [backend init](https://github.com/hashicorp/terraform/blob/main/internal/backend/init/init.go)
* [backend azure](https://github.com/hashicorp/terraform/tree/main/internal/backend/remote-state/azure)

**Homework**

* 嘗試修改，plan，apply，destroy 下列 resource
  + \_poc/container\_registry
  + \_poc/security\_group
  + \_poc/virtual\_network
* 透過 azure web console，container registry，使用 web console 更改，或是使用 terraform 更改設定
* destroy 以下內容以節省費用，我們目前不會再用到。當然同學還是可以自由練習
  + \_poc/container\_registry
* 選擇另一個 backend，並閱讀其說明文件
  + 如果手邊取得 backend，則依照設定試著改用這個 backend
  + 如果有問題，歡迎與底下留言

**Q&A**

請問我需要在每個不同的資料夾都設定 terraform.backend 嗎？感覺很不 DRY (don't repeat yourself)

沒錯，當使用 terraform 久了，很快就會發現這個問題，有沒有可能精簡 .tf 檔案呢？請見下堂課。

**References**

* <https://www.terraform.io/docs/language/state/backends.html>