**Remote module**

[Terraform 支援許多類型的 module](https://www.terraform.io/docs/language/modules/sources.html)

上一章使用的範例，module source 都是本地檔案的路徑，透過路徑去找到本機上的 module，稱為 local-path

module "registry" {

for\_each = toset(local.environments) # convert tuple to set of string

source = "../../..//azure/modules/container\_registry"

#source = "/Users/che-chia/my-workspace/terraform-30-days//azure/modules/contrainer\_registry"

...

}

除了 local path 以外，最常使用的是 Git

**Git remote module**

[官方文件在此](https://www.terraform.io/docs/language/modules/sources.html#generic-git-repository)

Git remote module 原理非常簡單，在 init 的時候，使用 git 向遠端的 git repository 取得 module .tf 檔案，在本地的 .terraform 裡頭暫存一份，plan 的時候使用 .terraform 內的 module。source 設定參考 \_poc/container\_registry\_module\_remote

module "registry" {

for\_each = toset(local.environments) *# convert tuple to set of string*

source = "git::ssh://git@github.com/chechiachang/terraform-30-days.git//azure/modules/container\_registry?ref=main"

*#source = "git::https://git@github.com/chechiachang/terraform-30-days.git//azure/modules/container\_registry?ref=main"*

*#source = "git@github.com:chechiachang/terraform-30-days.git//azure/modules/container\_registry?ref=v1.0.0"*

*#source = "git@github.com:chechiachang/terraform-30-days.git//azure/modules/container\_registry?ref=f1d8c86de3aebc40f16bc3a015f9a42b70dba209"*

...

}

* source 的格式 git::protocol://repo-url//path?ref=git-ref
  + git 支援使用 ssh / https，推薦使用 ssh
  + repo-url 就是 repository 完整 url，支援所有
  + 如果是私有的 repository 也可使用，使用 ssh key 存取十分方便
  + path 就是 module 在 repository 內的路徑 //azure/modules/...
  + (optional) git reference，可以指向 branch / git commit / tag
* 如果 repository 是使用 github.com / bitbucket，可以使用 github.com / bitcucket.org

**Git remote module behaior**

我們實際使用 \_poc/container\_registry\_module\_remote 來操作說明：

terraform init

Initializing modules...

Downloading git::ssh:*//git@github.com/chechiachang/terraform-30-days.git?ref=v0.0.1 for registry...*

- registry in .terraform/modules/registry/azure/modules/container\_registry

Downloading git::ssh:*//git@github.com/chechiachang/terraform-30-days.git?ref=v0.0.1 for test...*

- test in .terraform/modules/test/azure/modules/container\_registry

Initializing the backend...

...

Initializing provider plugins...

...

init 的時候，比起 local-path module，Terraform 額外執行 git clone，將 module 中的 source clone 到本地 .terraform 資料夾。照慣例進去開一下

tree -L 1 .terraform

.terraform

├── modules

├── providers

└── terraform.tfstate

2 directories, 1 file

cat .terraform/modules/modules.json | jq

{

"Modules": [

{

"Key": "test",

"Source": "git::ssh://git@github.com/chechiachang/terraform-30-days.git//azure/modules/container\_registry?ref=v0.0.1",

"Dir": ".terraform/modules/test/azure/modules/container\_registry"

},

{

"Key": "",

"Source": "",

"Dir": "."

},

{

"Key": "registry",

"Source": "git::ssh://git@github.com/chechiachang/terraform-30-days.git//azure/modules/container\_registry?ref=v0.0.1",

"Dir": ".terraform/modules/registry/azure/modules/container\_registry"

}

]

}

由於使用遠端的 module ，每次對 module 內杜設定有更改，都需要重新執行 terraform init，例如：

* 上一張提到的更改 module 名稱等設定
* 更改 source 使用其他 module
* 版本升級，使用新版的 module，更改 ?ref=v0.0.1 變成 ?ref=v0.0.2

除了需要 init 下載以外，terraform plan && terraform apply 都與之前的使用完全一樣。

**Git module Pros & Cons**

Pros

* 版本鎖定
* 方便分享，便利使用社群維護的 module
* 由於使用本地的 git 設定執行 git clone，自己電腦上有關 git 的設定與功能都會生效
  + i.e. private repository 也可以使用本地的 git credential 存取

Cons

* 增加 terraform init 需要的時間
* .terraform 使用更多硬碟空間
* 查找原始碼的時候，需要一層一層去搜尋多個 repository

目前我們的 module 結構簡單，而且沒有額外依賴其他 module，所以缺點感覺不太出來。實務上許多複雜的 module 內部都還會有更多曾依賴的 module，每個 module 自己還有依賴的 providers，每次下載都需要時間，.terraform 都會變一大包

雖然說了缺點，然而這些小缺點並不影響開發，我們還是會選擇使用 remote module

**Good Practices**

* 本地的開發與測試，可以使用 local-path，方便且加速流程
* 讓其他環境使用的 module 務必推到遠端，並且打上 version tag
* 永遠使用鎖定 git module 版本， tag > commit > branch
  + 避免遠端 branch 推進改變，影響本地
  + 使用 commit 的可讀性極差，會造成以後維護 pull 新版本的麻煩

**Terraform Registry**

Terraform 官方提供了 module 分享庫，稱為[Terraform Registry](https://www.terraform.io/docs/language/modules/sources.html" \l "terraform-registry" \t "_blank)，上面儲存許多 provider 與 modules

是的，當我們使用 provider {}，預設都會到 Terraform Registry 上去下載 provider 的檔案。Terraform 官方在上頭維護許多 remote module，可以直接使用。

直接看例子：

我們可以使用 [Terraform Registry: Azure/compute](https://registry.terraform.io/modules/Azure/compute/azurerm/latest)，來創建 Azure Compute VM，這個 module 除了 vm 以外，還包含常與 vm 共用的資源，例如 network interface，security group，public ip...等，貼近常見使用情境

azurerm\_availability\_set.vm

azurerm\_network\_interface.vm

azurerm\_network\_interface\_security\_group\_association.test

azurerm\_network\_security\_group.vm

azurerm\_network\_security\_rule.vm

azurerm\_public\_ip.vm

azurerm\_storage\_account.vm-sa

azurerm\_virtual\_machine.vm-linux

azurerm\_virtual\_machine.vm-windows

random\_id.vm-sa

使用上也非常方便，Terraform Registry 上多半會提供 Readme，範例，以及 input 的參數。

筆者個人比較少使用 Registry，筆者習慣直接看 Github 上的 .tf 程式碼，畢竟 provider 除了會將 module 上傳至 Terraform Registry 外，也會上傳到 Github，例如[Github: Azure compute](https://github.com/Azure/terraform-azurerm-compute" \t "_blank)

各位可以依照自己使用習慣選擇。也許之後 Terraform Registry 會推出更多新功能，期待之後的改進會變得更好用。

**Other**

Terraform 仍支援其他 remote module 如：s3, gcs...，筆者認為 git 還是最常使用的，而且其他類型 module 概念都類似，有需求的朋友可以依據需求嘗試使用其他 module。

**development with multiple environments**

理解 module 的基本原理與使用，接下來要說明實務中如何開發 module。

* 先 Google / Github 上找看看有沒有社群維護，寫好的 module 可以直接使用
* 為 module 與 .tf 準備測試環境，例如：
  + 先在 dev/southeastasia/container\_registry 編輯 local-path module
  + 穩定了，推上 branch / release candidate tag，讓 stag 使用。也會將 app 丟到環境中去測試 (stag/southeastasia/container\_registry)
  + QA 都測試完了，表示 module 已經穩定，打上 release tag，供 prod 使用(prod/southeastasia/container\_registry)
  + prod infrastructure 使用到的 tag 都會是穩定版本，依照穩定版本更新，避免不正確的 infrastructure 出現在 prod

module 是持續修改，很難一開始就寫出完美的 module。使用 terraform 來落實 infrastructure as code，可以在 infrastructure 中導入軟體開發流程，透過 gitflow，讓 instructure 方便測試，讓環境更穩定。

更多 Terraform IaC 的好處，我們在後面章節實際帶各位體會。

**Homework**

* \_poc/compute
  + 基本背景知識：需要對 Azure compute VM 有一定了解
  + 設定參數，參數的說明在[Github: azure/compute variables.tf](https://github.com/Azure/terraform-azurerm-compute/blob/master/variables.tf)，注意 default 值
  + 參考 [Github: azure/compute main.tf](https://github.com/Azure/terraform-azurerm-compute/blob/master/main.tf" \l "L32" \t "_blank)，調整參數
  + init && plan && apply
  + plan 產生較多 resource，請把它看完看清楚，養成好習慣
    - 不清楚的資源請在 google terraform + resource，查一下 resource 文件
  + 嘗試處理過程中的錯誤，一般來說設定正確的 variable 可以解決
  + 成功 apply 後，到 [azure web portal](https://portal.azure.com/)上看一下產生出來的資源
  + 修改參數，調整成自己喜歡的樣子
* 找另外一個 github 上的 remote module，嘗試參考 compute 的架構完成

NOTE:

* 請盡量使用 Standard\_B1s 免費的 vm size，詳請請見 [Azure Free Plan 清單](https://azure.microsoft.com/zh-tw/free/search?WT.mc_id=AZ-MVP-5003985)
* 完成後執行 terraform destroy，清除所有資源，以節省費用

**Can it be more DRY**

* 每個 root module 都有 provider.tf
* 重複的 locals 參數，例如：resource\_gropu\_name，location，environment...
* provider.tf 中需要手動指定 backend key = "xxx.tfstate"，才能將各自的 state 保存在遠端獨立的路徑

backend "azurerm" {

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

storage\_account\_name = "tfstatee903f2ef317fb0b4"

container\_name = "tfstate"

key = "container\_registry\_module.tfstate"

}

有沒有可能在更精簡，更方便？ 我們介紹 Terragrunt