手動上 web console / portal 點擊的步驟很難自動化，但使用 IaC 工具（ex. terraform，ansible，chef，salt...) 才容易執行自動化

自動化有許多特點

* 工程師的工作，從手動 apply infrastructure，變成維護自動 apply infrastructure 的 workflow
  + 重複的工作變少，效率變高
  + 困難且複雜的工作變多（是的，人工智慧比工人智慧難多了）
* 降低人為失誤（human error）

這樣有個迷思，說自動化就是好棒棒。自動化不是萬靈丹，套用自動化，團隊就永遠不出錯。試想：workflow 還是會寫錯，萬一錯的是 workflow，就會不斷的做錯

* IaC 只是記錄 workflow，並逐漸迭代改進 workflow
  + 固定現在的流程
  + 快速復現（reproduce）workflow 的錯誤
  + fix，commit，Review，apply 新流程
* 另一個文化上的體現是：工程師做錯不是處罰工程師，而是團隊一起修復 SOP / workflow

此外，本篇使用 Github Action 做操作，然而絕大多是成熟的 CI 工具，都可以做到底下描述的內容，團隊可以自由選擇 CI 工具，例如：Jenkins，CircleCI ...都會是很好的選擇

[Terraform 官方的 automate 指南](https://learn.hashicorp.com/tutorials/terraform/automate-terraform) 使用 Terraform Cloud，我們這邊先不使用 Terraform Cloud，而是使用 Github Action。然而基本的工作流程與官方文件描述相近。

**Config Github**

上篇[IAM for terraform](https://ithelp.ithome.com.tw/articles/10262124" \t "_blank) 我們為 Terraform 設定獨立的 service principal，並取得 credential（certtificate in .pfx）作為認證的 credential。只要把這些 secret 上傳 Github Secret，搭配 terraform 的 binary，理論上就能在 Github Action 執行所有 Terraform 操作，控制 Azure resources。需要的環境變數如下：

export ARM\_CLIENT\_ID="00000000-0000-0000-0000-000000000000"

export ARM\_CLIENT\_CERTIFICATE\_PATH="/Users/che-chia/.ssh/terraform-30-days.pfx"

export ARM\_CLIENT\_CERTIFICATE\_PASSWORD=<password>

export ARM\_SUBSCRIPTION\_ID="00000000-0000-0000-0000-000000000000"

export ARM\_TENANT\_ID="00000000-0000-0000-0000-000000000000"

需要上傳一個檔案，指定五個環境變數。細節請見[Github 文件：create secret](https://docs.github.com/en/actions/reference/encrypted-secrets" \l "creating-encrypted-secrets-for-a-repository" \t "_blank)

* 到 Github Repository -> Settings -> Secrets -> New Repository Secret
* 將 .pfx 轉成 base64，用 text 格式貼到 secret，命名為 ARM\_CLIENT\_CERTIFICATE\_BASE64
* 將其他參數依照原本的名稱上傳 secret
* Github 會將上述 secret 都加密

由於 Github Secret 不支援檔案上傳，所以我們將 .pfx 轉成 base64，用 text 格式貼到 secret，命名為 ARM\_CLIENT\_CERTIFICATE\_BASE64，在 Github Action 使用前，做 base64 -d 解開成為原來的檔案。注意 base64 輸出應該是一行內容，貼上時不要有斷行符號

cat ~/.ssh/terraform-30-days.pfx | base64

MIIPeQIBAzC....................

...

...

................QICCAA=

實際上傳到 Secret 的參數

ARM\_CLIENT\_ID="00000000-0000-0000-0000-000000000000"

ARM\_CLIENT\_CERTIFICATE\_BASE64="..."

ARM\_CLIENT\_CERTIFICATE\_PASSWORD=<password>

ARM\_SUBSCRIPTION\_ID="00000000-0000-0000-0000-000000000000"

ARM\_TENANT\_ID="00000000-0000-0000-0000-000000000000"

**Config workflow**

在 github workflow 中，[參考官方文件](https://docs.github.com/en/actions/reference/encrypted-secrets" \l "using-encrypted-secrets-in-a-workflow" \t "_blank)，就可以在 workflow.steps 中取用

steps:

- name: Hello world action

with: *# Set the secret as an input*

super\_secret: ${{ secrets.SuperSecret }}

env: *# Or as an environment variable*

super\_secret: ${{ secrets.SuperSecret }}

更改 .github/workflow/plan.yaml

* <https://github.com/hashicorp/setup-terraform>

yq read .github/workflows/plan.yaml

使用 [nektos/act 工具測試 Github Action](https://github.com/nektos/act" \t "_blank)

sudo port install act

act *--version*

act version 0.2.24

本地測試 .github/workflow

* 做 github action 格式檢查，避免上傳錯誤的 yaml，還要等 Action 執行才發現
* 分項，dry-run，debug
* 本地測試

act *--list*

ID Stage Name

terraform-fmt 0 trerraform fmt

terraform-plan 0 Terraform Plan

validate 0 Validate terraform configuration

執行 Dry run，沒有實際讓 Action 實際運行，只是把 .yaml 喂進去，確定沒有 syntax error

act *--dryrun*

測試觸發 push event 時，github Action 會執行的 job

* (Optional) act 可以帶入額外的參數，帶入 secret env 做測試會更準確
  + 需要明碼的 azure credential，包括 .pfx, password, ...，工程師本機也不應該可以取得這些 secret
* 實務上，把不需要 credential 的部分測一測就可以推上去 Github Action 測試了

act *--env foo=bar*

**Workflow for Terraform**

Terraform 的 CI/CD 流程有非常多的面相

* push / PR checks
  + format / validate
  + module test
* security checks
* cost checks
* terraform functions
  + init
  + plan
  + apply

tree .github

.github

└── workflows

├── format.yaml

├── plan.yaml

├── security-scan.yaml

└── validate.yaml

本 repository 並沒有準備自動 apply 的 Github Action

* 本 repository 的 module 都是課程的範例，並不會對應實際我個人 azure cloud 上的狀況。簡單說：我不希望  
  terraform-30-days 的 module 都 apply 上去
* 如果是管理公司實際環境的 terraform repository，建議就可以啟用 terraform apply，on master push event

設定可以參考 .github/workflow/plan.yml 這份，調整一下 command 就可以

關於 Iac code 自動 apply，下一章 atlantis 我們會細講

**References**

* <https://github.com/hashicorp/setup-terraform>

**Homework**

* 在 .github/workflows/ 中建立屬於自己團隊的 workflow.yaml
* 閱讀 [Github Action: starter example](https://github.com/actions/starter-workflows" \t "_blank) 中的範例 workflow，套用已經寫好的 workflow 到自己的 repository 中