

← Back to index



Phil Huang
Microsoft Senior
Cloud Solution
Architect | CNCF
Ambassador

Metadata

📅 20230301

📁 in linux

Table of contents

前言

宣告式 (Declarative) 與命令式 (Imperative)

來個例子: 以部署 Azure Red Hat OpenShift

平台分析和個人工具選擇表

Azure

關於 1. 平台部署 (Infrastructure Provisioning) 階段

關於 2. 平台管理 (Infrastructure Configurations) 階段

VMware vSphere

Kubernetes

Linux 作業系統

Windows 作業系統

TL;DR

Q&A

Q1: 要研究一個新服務的時候，我應該要先用 Shell Script 寫還是 Ansible 模組呢？

Q2: Ansible 既然什麼都可以做，那是不是都不用學 Shell Script？

Q3: 針對管理 Azure，Ansible 既然有 ansible.azcollections，那為什麼我還需要用 Terraform 來部署呢？

Q4: 如果只有一套 IaC 要選，選哪套？

Q5: 針對 Azure，起碼有 3 個不同的工具可以選擇 Azure CLI、Terraform、Ansible，那我要怎麼選？

Q6: 題外話 docker-compose 算不算 IaC 實踐的一環？

Q7: 我不會寫 Terraform / Ansible / Shell Script，那要怎麼寫？

Q8: 你這麼推 Ansible，那 Puppet / Saltstack / Chef 情何以堪？

References

linux iac

選擇 IaC 工具是多選題，而不是單選題



前言

很多朋友講到 Infrastructure as Code (IaC) 的時候多半都會直接落在單一工具選擇的討論，想要窮舉出最終選擇，但隨著時間久了我覺得這個議題應該要以共生共榮的角度看待比較好

早期的時候都是 Shell Script 當道居多，直至 2012 年左右 Ansible (2015 年被 Red Hat 收購)、Puppet (2022 年被 Perforce 收購)、Chef (2020 年被 Progress 收購)、Saltstack (2020 年被 VMware 收購) 等 4 大多數以作業系統管理為主的組態管理 (Configuration Management) 興盛，到各大雲廠商迅速崛起出現 Hashicorp Terraform (2021 年 IPO) 這類以雲平台 (Cloud Provider) 為 IaC 核心的工具，具體來講，被管理的平台不同，選擇的方向會有點不一樣，下面會有一些我個人的經驗分享

宣告式 (Declarative) 與命令式 (Imperative)

絕大部分的 IT 管理人員應該都是比較習慣命令式 (Imperative) 的操作方式，畢竟誰不是用 Bash、CLI 長大的呢？但隨著服務越來越多和已知的資源相依性問題，後來漸漸地宣告式 (Declarative) 管理成為主流，其實最經典的案例就是 Kubernetes，但其實你不太需要了解到這麼細，反正都是要管理，用最有利於團隊的工具，達到最終目的就好

方式	解釋	優點	缺點
命令式 (Imperative)	明確陳述為產生所需結果所執行的命令	1. 對於操作偏向 Step by Step 性質的相當好用 2. 對於一次性工作好用 3. 所有系統管理的基本功	1. 平台層級的相依性處理不強 2. 需要一定的 Coding 技能，如 Shell Script、Powershell 等，處理錯誤和 if-else 判斷 3. 設定飄移很難修正，多數需要個別處理
宣告式 (Declarative)	指定想要的結果，而不是指定要如何完成	1. 不太需要 Coding 技能即可使用 2. 高度重複性 3. 容易修正設定飄移問題	1. 對於流程控制較不彈性 2. Day 2 Operation 不強，譬如說既有 OS 改 IP

來個例子: 以部署 Azure Red Hat OpenShift

因今年來 Microsoft Azure 上班，配上剛好之前在 Red Hat 上班的經驗，我下面主要會以 Azure Red Hat OpenShift 部署經驗為主，該撞到的地雷我全部都撞到了，分享我個人在選擇 IaC 工具時的一些經驗

工具	屬性	資源管理方式	可讀性	可擴展性	可維護性	管理顆粒度	優點	缺點
Azure CLI	命令式 (Imperative)	Azure Resource Manager	●	●	●	●	指令相當直覺，一次性的操作好用	Shell Script 動不動就一百行起跳，有考慮變數、if-else 判斷、Error Handling 的話會更複雜
Ansible Playbook	宣告式 (Declarative)	Azure Resource Manager	●	●	●	●	萬用膠語言	需要先透過 Azure CLI 一步一步了解操作流程才能寫出 Ansible Playbook
ARM Template	宣告式 (Declarative)	Azure Resource Manager	●	●	●	●	一次性的操作好用，可支援 Azure Portal 上呈現需部署欄位	我個人覺得難讀和難改
Terraform	宣告式 (Declarative)	Azure Resource Manager	●	●	●	●	資源相依性處理強，部署 Cloud Provider 資源首選	需要知道 Terraform HCL 怎麼寫，這對蠻多人是個門檻的


這邊針對部署 Private Azure Red Hat OpenShift 提供參考範例，[pichuang/compare-iac-implementation](#)

平台分析和個人工具選擇表

基於不同平台之下，下面提供我個人知道，且用過的 Infrastructure as Code 的排列組合

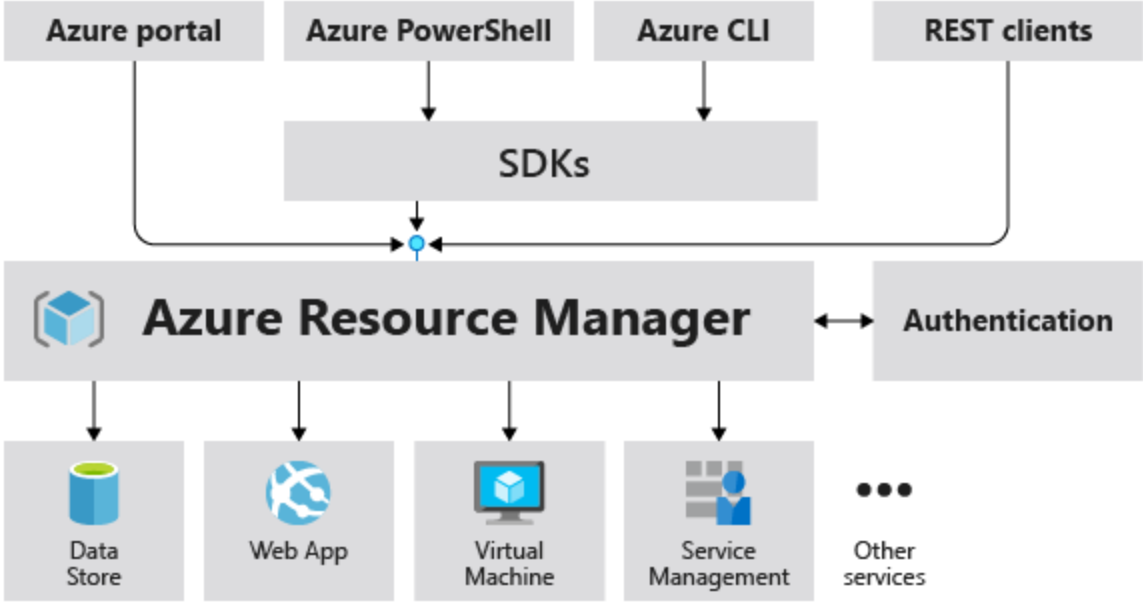
Platform	Consistent Management Layer	GUI	CLI	+ Ansible CLI Collections	Ansible Collections	Terraform P
Azure	Azure Resource Manager	Azure Portal	Azure CLI / Azure PowerShell	ansible.builtin.shell	ansible.azure.azcollection	hashicorp/az
VMware vSphere	VMware vCenter (vc)	VMware vCenter WEB UI	govc / VMware PowerCLI	ansible.builtin.shell	ansible.community.vmware	hashicorp/vs 人少用
Kubernetes	Kubernetes API Server	N/A	kubectl	ansible.builtin.shell	ansible.kubernetes.core	hashicorp/ku 個人少用
Red Hat OpenShift	Kubernetes API Server	OpenShift Web Console	oc / kubectl	ansible.builtin.shell	redhat.openshift, 個人少用	N/A
VMware vSphere with Tanzu	VMware vCenter 7 (vc) + Custer API	VMware vCenter	kubectl vsphere	ansible.builtin.shell	N/A	N/A
VMware Tanzu Kubernetes Grid 1.x	Kubernetes API Server + Custer API	N/A	tanzu / kubectl	ansible.builtin.shell	N/A	N/A
Linux	N/A	X Windows	bash	ansible.builtin.shell	Depends on Linux Distribution	N/A
Windows	N/A	Windows Desktop	PowerShell	ansible.windows.win_command	ansible.windows.win_command	N/A
Cisco 2960 Switch	N/A	N/A	IOS CLI	N/A	ansible.cisco.ios	N/A

Azure

 **Azure iac 對策**

先 Terraform，後 ansible.builtin.shell + Azure CLI

基於 [What is Azure Resource Manager?](#) 所述，Azure Resource Manger 提供了如下圖所呈現的統一管理層 (Consistent Management Layer)，提供統一的 API，無論你是使用 Azure Portal (屬於 ARM Template 範疇)、Azure PowerShell、Azure CLI 或者用 Postman 自己慢慢打 REST API，其實都是呼叫到底層相同的 API Call，理論上並不會有不相同的狀況發生，但實務上你有時會發現上面幾個工具操作有點不一樣，主要原因在於 Azure Resource Manager 上面所提供 SDK 可能還沒實作相關功能、API Version 還沒升上去或者是根本上呼叫操作的時候是考慮特定狀況而寫死，這些都算是常見的狀況



針對 Azure Cloud Provider 部屬和管理上，可以分為 2 個階段:

- 平台部署 (Infrastructre Provisioning): 從 0 到 1, 服務從無到有，包含網路、資料庫、虛擬機 VM Size 等等
- 平台管理 (Infrastructure Configurations): 從 1 到 2, 既有服務持續維護，不做平台層級的變更

關於 1. 平台部署 (Infrastructre Provisioning) 階段

選擇上會使用到的工具組合有

- 單獨使用 Terraform 所提供的 azurerm
- 以 Azure CLI 功能為主，Ansible 或 Shell Script 為輔助管理操作流程

預設狀況下，你應該都要用 Terraform 完成從無到有的平台部署階段，讓 AzureRM Terraform Provider 直接對 Azure Resource Manager 進行管理，文件可參考 [hashicorp/terraform-provider-azurerm](#)，故把 Terraform 寫好的確可以加速平台部署速度和管理，但...有時候你會遇到 Terraform 所提供的功能並非是完善的，譬如說 Azure Red Hat OpenShift (aro) 的部署能力直至今日還在 [hashicorp/terraform-provider-azurerm - New Resource: azurerm_redhat_openshift_cluster #20266](#) 躺著，你用官方 azurerm 是寫不出來相關功能的，這時候你就需要換下一個做法了

Azure CLI 跟 Azure PowerShell 畢竟是 Azure 親兒子，相關指令和測試都寫得相當完善，如果你看到新的功能出的時候，多半最新的 Azure CLI 也都有提供對應的指令給你操作了，所以只要遇到太新的服務，Terraform 測不出來，你可以直接換成用 Azure CLI 為主的方式來操作，輔以 Shell Script 或者是 Ansible 來完成整體的操作流程，如果你是單純用 Bash + Azure CLI 的話，風格如範例 [Azure CLI + Bash - azcli-create-private-aro.azcli](#)

為何不使用 ARM Template 或 Bicep? 我個人覺得是很難寫，很難將具備邏輯和流程相關的元素寫進去，但一次部署的可以用，但這個太容易被 Terraform 角色取代，或者是忘記將功能寫進去，看看 Azure Red Hat OpenShift 在 Azure Portal 上你是點不出 Private 架構的...除非你自己手動匯入 ARM Template，如範例 [ARM Template - arm-template-create-private-aro.json](#)

為何不使用 Anisble Module 提供的 [Ansible - Azure.Azcollection](#)? 這個要看狀況，多數不使用的理由跟 Terraform 有些 Azure 服務真的沒實作出來一樣，抑或者是該模組太久沒更新，以 Azure Red Hat OpenShift 為例，
[azure.azcollection.azure_rm_openshiftmanagedcluster](#) 是 Ansible Module 名稱，雖然模組有是有，但有段時間沒更新了，導致某部分 API 呼叫會失敗，實務上部署不出來，所以最終還是用 Azure CLI 來完成操作。

但我並不是否定 azure.azcollection 所有的模組，只要是常見的服務，如網路、部署虛擬機等等，使用上都不會有太大問題，所以正確用法是 Azure Collection + Azure CLI 混著用，而不是只用單一個選擇，風格如範例 [Ansible Playbook - ansible-create-private-aro.yml](#)，常用服務用 azure.azcollection 模組，不常用或新服務用 Azure CLI + Ansible Module - shell, command 來完成整體的操作流程

關於 2. 平台管理 (Infrastructure Configurations) 階段

分為兩個部分看，如果你是管理 PaaS 之類的服務，的確你還是可以繼續使用 Terraform 或者是用 `az xxx update` 來進行基於平台的設定更新管理，但如果是 VM 層級的服務，需要動到作業系統裡的設定，如 Ubuntu 想要改 Interface IP, Red Hat Enterprise Linux 需要上 `sysctl`、Windows Server Firewall Allow ICMP 等等偏向作業系統層級的管理，沒意外就是以 Ansible 為主了，Ansible 最有優勢的地方就是 Agentless 這個特性，有 SSH 或 WinRM 就能做後續的作業系統管理了，詳細可以參考小弟之前寫的 [30 分鐘內從開始到入門的 Ansible](#)

VMware vSphere

 VMware vSphere Iac 對策


首選 Ansible 並大量使用 `ansible.community.vmware`

雖然 VMware vSphere 是個平台，也有 Terraform Provider 的模組，但實務上如果你不是把它當作 Cloud Provider 的使用方式的話，偏向是一步一步操作的話，可能用 VMware vCenter Web UI 操作居多

倘若你一定要做 IaC 又以 Ansible 為主的話，你一定會大量用到 [community.vmware.vmware_guest](#)，無論是自動化從 VM Template 部署一台新的 VM、針對指定機器開關機、修改 VM Instance，但這模組管理的層級理論上是不會深入到 OS 裡面的，主要都是以 VMware VM instance 角度再做管理，並不涉及到作業系統內部的設定修改，跟 `az vm` 和 `govc vm.clone` 有異曲同工之妙

Kubernetes



 Kubernetes Iac 對策

針對 Kubernetes 裡面的資源對策 `ansible.builtin.shell` + `kubectl/helm`
針對 Kubernetes 外部的服務對策，回歸到各自平台的管理方式，如 VMware vSphere、Azure、AWS、GCP 等等

Kubernetes 平台同時間包含了平台管理和應用程式部署，最沒問題的部分就是應用程式部署，事情最單純，你找哪一家或者是哪一座都差不多處理方式，最大的議題是在 Kubernetes 平台管理上

管理 Kubernetes 平台有 2 個層面要討論

- 在 Kubernetes 裡面的資源 (Resource): 主要用 `kubectl/helm` 指令操作即可，可以搭著 Ansible [ansible.buitin.shell](#) 來做，這個事情也算小，基本上跟應用程式部署差不多層級，只是比較偏向 Kubernetes 內部視野的平台管理，CNCF CKA / CKAD / CKS 基本上都是考這個範圍的知識

2. 在 Kubernetes 外面的服務 (Services/Provider): 主要是無法用 kubectl 操作皆屬此類，大多數議題都跟 Kubernetes 議題無關，如 Underlay Networking、節點大小、開關機、External Load Balancer 介接、NetAPP Trident CSI 串接、F5 BIG-IP CIS 整合、Veritas Kubernetes 備份等等，這個是最具挑戰的地方，因為很吃既有方案對接 Kubernetes 整合的理解

2 個問題可以讓你分清議題


- 最核心問題是 你的 Kubernetes 是誰家的? 這個會影響到你要如何管理 Kubernetes，如公有雲提供的 Kubernetes，如 AKS、EKS、GKE 等，還是廠商提供為主的 Kubernetes，如 Red Hat OpenShift on Bare Metal、Red Hat OpenShift on VMware vSphere、VMware vSphere with Tanzu、VMware Tanzu Kubernetes Grid、Rancher、Rancher 等，或者是自架 DIY 的 Kubernetes
- 其次是 Cluster API 有沒有用? 有用的話，Kubernetes 可以透過 Cluster API Provider 的方式去調度底層資源，架構設計得當的狀況下你不太需要插手，可參考 20211122_Kubernetes 多叢集及單叢集架構選擇探討，沒用的話就是回歸到各自底層平台的管理方式，地端我遇到的案子基本上都是沒用的

Kubernetes Solution	Platform	Cluster API Enabled?	Kubernetes 外部服務資源管理方式	Kubernetes 內部資源管理方式
Azure Kuberentes Service	Azure	✔	1. az aks 2. Terraform	kubectl/helm
Red Hat OpenShift 4	Bare Metal	✔	ansible.Dellemc.Openmanage 或者是 HPE iLo, 這個比較特殊一點	kubectl/oc/helm
Red Hat OpenShift 4	Bare Metal	✖	ansible.Dellemc.Openmanage 或者是 HPE iLo	kubectl/oc/helm
Red Hat OpenShift 4	VMware vSphere	✔	VMware vCenter	kubectl/oc/helm
Red Hat OpenShift 4	VMware vSphere	✖	ansible.community.vmware	kubectl/oc/helm
Red Hat OpenShift 4	Azure	✔	Terraform	kubectl/oc/helm
Azure Red Hat OpenShift	Azure	✔	Azure CLI	kubectl/oc/helm
VMware vSphere with Tanzu	VMware vSphere	✔	VMware vCenter	kubectl/kapp
VMware Tanzu Kubernetes Grid 1.x	VMware vSphere	✔	tanzu	kubectl/kapp
VMware Tanzu Kubernetes Grid 1.x	Azure	✔	tanzu	kubectl/kapp

針對 OpenShift 4 內部資源管理方式，我之前有針對 OpenShift Day 2 Operations 常遇到的管理情境，用 Ansible 寫了不少範例下來可以參考 pichuang/openshift4-toolbox，例如 EtcD 備份、檢查、叢集開關機、安裝 Service Mesh 皆能寫

另外 VMware 相關的解決方案 VMware vSphere with Tanzu 和 VMware Tanzu Kubernetes Grid，會相當著重在 VMware vSphere 和 Tanzu Management Cluster (mc)/Supervisor Cluster(sc) 的架構設計上，但這個超出討論範圍，本文不討論

Linux 作業系統

 Linux IaC 對策

ansible.builtin.shell + Ansible Collections + Shell Script


太多了講不完，網路上參考文章非常多，如果不嫌棄的話可參考 30 分鐘內從開始到入門的 Ansible 和 Ansible 技術概觀介紹_20190130，主要先會 2 個就好，其他都是變化而已

```
# 1: 對多個伺服器下相同指令
ansible -i inventory all -m shell -k -a <指令>

# 2: 將一個檔案或資料夾傳送到多個伺服器上
# 檔案
ansible -i inventory all -m copy -k -a "src=/etc/motd dest=/etc/motd"
# 資料夾
ansible -i inventory all -m copy -k -a "src=/etc/yum.repos.d/ dest=/etc/yum.repos.d/"
```

為何不使用 Terraform? 普遍 Linux 作業系統都沒有 API 可以讓你呼叫，所以也自然沒有相關模組可以用，都是滿滿的 CLI...

Windows 作業系統

 Windows IaC 對策

ansible.windows.win_command + Ansible Collections + PowerShell Script

採用邏輯跟管理 Linux 作業系統一樣，這邊提供幾個我常用的範例

- 針對 Windows Firewall 開 ICMP request
- 傳特定檔案到所有 Windows Server 上

只是因為 Windows 權限預設開得很低，在一開始使用的時候需要先特別把 Execution Policy 設定成 RemoteSigned，這樣才能讓 Ansible 透過 WinRM 來執行 Powershell Script

set-remotesigned-policy.ps1

Set-ExecutionPolicy RemoteSigned
Get-ExecutionPolicy

winrm quickconfig

Enable WinRm Auth
winrm set winrm/config/service/auth '@{Basic="true"}'

Enable Allow Un-encrypt
winrm set winrm/config/service '@{AllowUnencrypted="true"}'

Verify WinRm Listener
winrm enumerate winrm/config/listener

為何不使用 Terraform? Windows 在作業系統層級上雖說有一票 API 可以用，但現行沒人針對 Windows 撰寫屬於他的 Terraform Provider，我相信以後也不會有。但如果你是說 Hyper-v 開起來的狀況，那...這個就是另外一個討論了，現行兩邊官方是沒支援啦...

TL;DR

- 1. Terraform 搭平台根基
- 2. Ansible 管肚子和黏服務
- 3. GitHub Copilot + OpenAI ChatGPT 上雙 Buff

Q&A

Q1: 要研究一個新服務的時候，我應該要先用 Shell Script 寫還是 Ansible 模組兜?

Shell Script 先寫，因為你會有很大時間在研究到底怎麼把 CLI 兜起來，等到兜起來之後，如果有需要分享給其他人執行的話，可以改寫成 Ansible Playbook，會更具可讀性和維護性

Q2: Ansible 既然什麼都可以做，那是不是都不用學 Shell Script?

不，Ansible 強項是 Step by Step 流程控制，但對於文字處理和檔案處理我覺得超級弱，譬如說你用 Ansible 處理 JSON/YAML 格式轉換保證轉到吐血

解法是用 ansible.builtin.shell 直接呼叫 Shell Script，然後那個 Shell Script 可以在裡面用你常用的工具做檔案處理

- name: Convert JSON to YAML using Shell Script
 become: true
 vars:
 json_file: /path/to/input.json
 yaml_file: /path/to/output.yaml
 shell: covert_json_to_yaml.sh {{ json_file }} {{ yaml_file }}

Q3: 針對管理 Azure，Ansible 既然有 ansible.azcollections，那為什麼我還需要用 Terraform 來部署呢?

Ansible 並不具備資源相依性處理能力，如果你部署 AKS 服務，但你沒有先把 VNet 之類的服務先弄好，他是沒辦法知道這件事，但 Terraform 可以，所以如果你要用 Ansible 完成類同於 Terraform 的事情，你勢必要花不少時間用 Azure CLI 先理解每一步資源是怎麼被建立起來的，然後再用 Ansible 來實作，時間成本不太合，

故個人建議可以定調

- 1. 平台部署 (Infrastucture Provisioning): Terraform
- 2. 平台管理 (Infrastructure Configurations): Ansible

Q4: 如果只有一套 IaC 要選，選哪套?

Ansible 吧，Agenless 特性可以很簡單把一堆奇怪服務像膠水一樣黏在一起，是一個萬能膠等級 DSL

Q5: 針對 Azure，起碼有 3 個不同的工具可以選擇 Azure CLI、Terraform、Ansible，那我要怎麼選?

先 Terraform，寫不出來用 Azure CLI 硬上看看，如果要將過程自動化，再用 Ansible 包起來

主要原因是因為模組更新頻率是 REST API > Azure CLI > Terraform >>> Ansible Collections，但多數狀況的使用場景都是常見的服務，所以可以以 Terraform 為主，大多數都有支援

Q6: 題外話 docker-compose 算不算 IaC 實踐的一環?

Yes. 可預測結果、可重複流程、可版控程式碼，是一個很好的 IaC 實踐

Q7: 我不會寫 Terraform / Ansible / Shell Script，那要怎麼寫?

找廠商服務 (x)，GitHub Copilot 用起來 (o)

GitHub Copilot 背後技術是由 OpenAI 所提供的 Codex 模型所支援的，而 Codex 模型是基於 GPT-3 所訓練出來的，GPT-3 是一個自然語言處理模型，可以根據輸入的文字或 Comments，自動產生程式碼，相當強悍好用

但實際上你還是要對這些語言本身要有一點理解，不然會很難判斷 GitHub Copilot 寫出來的東西到底是對還是錯

如果真的想要有個 24hr 小老師教學的話，就是要用 OpenAI ChatGPT 了

Q8: 你這麼推 Ansible，那 Puppet / Saltstack / Chef 情何以堪?

我個人太吃 Agentless 設計了，其他三家因為都需要塞 Agent 進去，所以很早期就放棄不太看

References

- [What is Azure Resource Manager?](#)
- [hashicorp/terraform-provider-azurerm](#)
- [New Resource: azurerm_redhat_openshift_cluster #20266](#)
- [Ansible - Azure.Azcollection](#)
- [azure.azcollection.azure_rm_openshiftmanagedcluster module – Manage Azure Red Hat OpenShift Managed Cluster instance](#)