

# 一行指令搞定所有伺服器環境設定

Allen



# 目錄

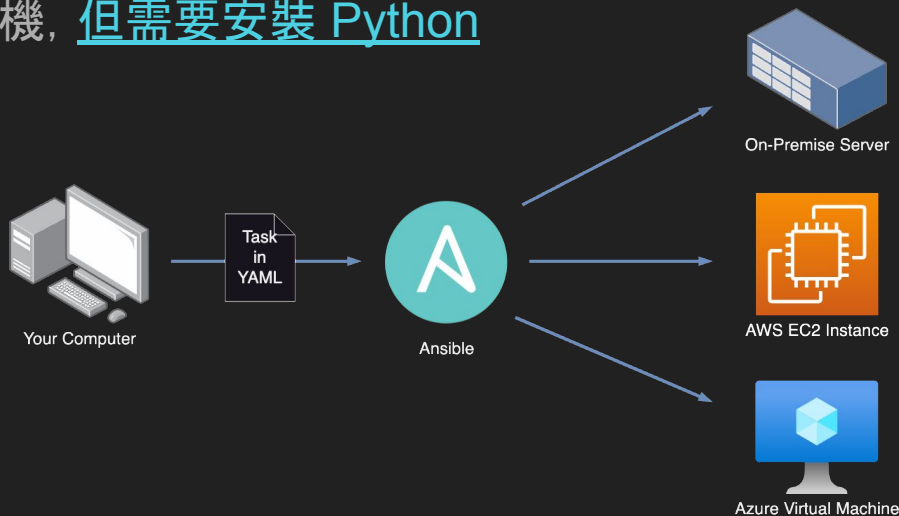
- What is Ansible?
- Inventory
- Modules
- Playbooks
- Idempotence (冪等性)
- Variables
- Conditionals
- Roles
- Demo

# What is Ansible?

Ansible 是一套知名的 IaC (Infrastructure as Code) 工具

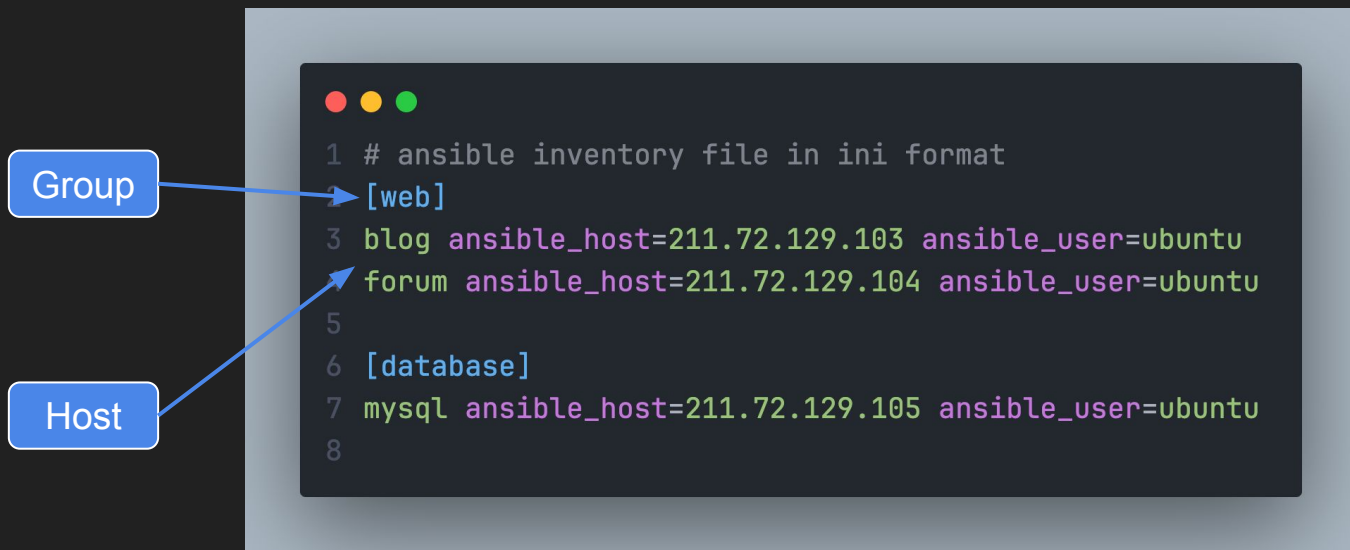
有別於用來建立資源的 Terraform, Ansible 主要用在設定機器上的環境

Ansible 最大的特點在於它是 **Agentless** 的, 你不需要在遠端主機上安裝任何 agent 就能操作遠端主機, 但需要安裝 Python



# Inventory

是一個用來定義主機連線資訊的檔案。Ansible 會參考該檔案並使用 SSH 連線至遠端主機



P.S. Group 與 host 的名稱請使用蛇形命名。ex. hello\_world、web\_server

# Inventory in YAML (鴨 Mo)

除了 INI, inventory 也可以用 YAML 格式書寫

```
1 # ansible inventory file in yaml format
2 web:
3   hosts:
4     blog:
5       ansible_host: 211.72.129.103
6       ansible_user: ubuntu
7     forum:
8       ansible_host: 211.72.129.104
9       ansible_user: ubuntu
10 database:
11   hosts:
12     mysql:
13       ansible_host: 211.72.129.105
14       ansible_user: ubuntu
15
```

# 使用 SSH Config 搭配 Inventory

Inventory 可以搭配 `~/.ssh/config` 使用

```
1 Host blog
2   User ubuntu
3   Hostname 211.72.129.103
4   Port 22
5   IdentityFile ~/.ssh/id_ed25519.pub
6
7 Host forum
8   User ubuntu
9   Hostname 211.72.129.103
10  Port 22
11  IdentityFile ~/.ssh/id_ed25519.pub
12
13 Host mysql
14   User ubuntu
15   Hostname 211.72.129.105
16   Port 22
17   IdentityFile ~/.ssh/id_ed25519.pub
18
```

```
1 # with ssh config file
2 web:
3   hosts:
4     → blog:
5     → forum:
6   database:
7     hosts:
8     → mysql:
9
```

# 連線測試

設定好 Inventory, 讓我們來使用指令測試一下

- **all**: 指定要連線的目標名稱, 可以是 group, 也可以是 host。這裡 **all** 代表全部
- **-i**: 指定要使用的 inventory 檔案
- **-m**: 指定要使用的 Ansible module



```
1 ansible all -i inventory.yaml -m ping
```

# Module

Ansible 擁有許多不同功能的 module 可以使用, 例如剛剛使用的 [ping](#) 就是一種 module, 其他還包含

- [apt](#): 管理主機上 apt 套件 (適用於 Ubuntu or Debian)
- [command](#): 在主機上執行指令
- [file](#): 管理主機上的檔案內容
- [fetch](#): 從主機上拉取檔案到本機

還有非常多 ...



# Playbooks

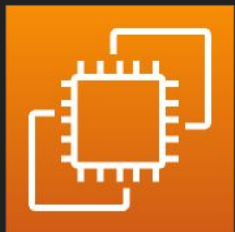
Playbooks 讓你可以多部主機上執行任務

甚至可以在多部主機上執行**有順序性**的任務 (A 做完換 B, B 做完再換 C)

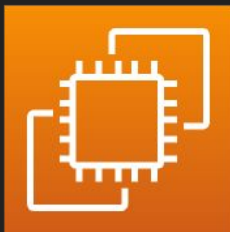
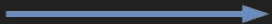
- Install Mysql
- Update the security settings
- Allow 3306 TCP Port in Firewall

- Install Laravel App
- Update Laravel settings
- Create the Database Schema

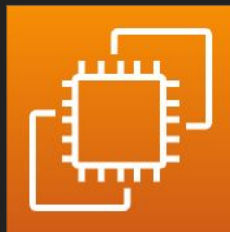
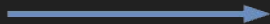
- Install Nginx
- Update the Nginx Config
- Allow 443 TCP Port in Firewall



Host A



Host B



Host C

# Playbooks 範例

Playbooks 為 YAML 格式

**hosts**: 定義在 inventory 中的 host 或是 group

**name**: 任務描述, Ansible Lint 會要求大寫開頭

**module**: 模組, Ansible Lint 會要求使用 fqcn (fully-qualified collection names) 的方式書寫

**parameter**: 模組的參數

```
1 - name: Install nginx and enable it
2   hosts: webserver
3   become: true
4   tasks:
5     - name: Install nginx
6       ansible.builtin.apt:
7         name: nginx
8         state: present
9     - name: Start the nginx service and enable it
10      ansible.builtin.service:
11        name: nginx
12        state: started
13        enabled: true
```

# 執行 Playbooks

使用 `ansible-playbook` 執行你寫好的 YAML 檔案



```
1 ansible-playbook playbooks.yaml -i inventory.yaml
```

# 每次都要指定 Inventory 好麻煩～

可以建立一個 `ansible.cfg` 檔案。這是 Ansible 的設定檔案，有[許多設定](#)可以設置

例如設定好 inventory 的檔案位置，之後輸入指令都不需要 `-i inventory.yaml`



```
1 [defaults]
```

```
2 inventory = ./inventory.yaml
```

# 先等等，這個 Shell Script 就能做到了吧？

你剛剛的 playbooks 範例，我兩行 shell script 就能搞定



```
1 #!/bin/bash
```

```
2
```

```
3 sudo apt install nginx -y
```

```
4 sudo systemctl enable nginx
```

# Idempotence (冪等性)

執行同一個動作一次或多次，被執行對象的結果狀態會是一致的

冪等性是寫任務時一個很重要的觀念

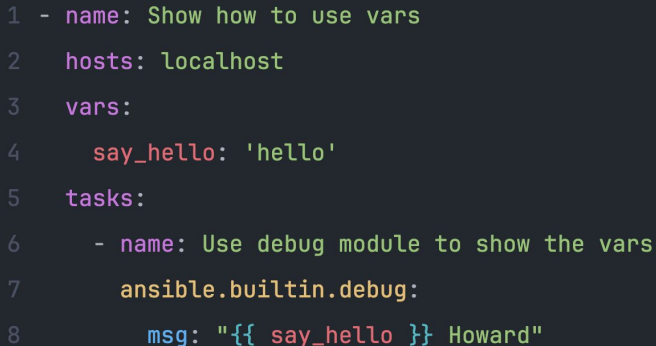
```
1 # it's ok to execute it multiple times
2 # this task will make sure the file "./foo.txt" is absent on the machine
3 - name: Install nginx and enable it
4   hosts: localhost
5   tasks:
6     - name: Remove file (delete file)
7       ansible.builtin.file:
8         path: ./foo.txt
9         state: absent
```

```
1 # each execute will append a line to the file
2 # so it's not an idempotent task
3 - name: Append a line to a file
4   hosts: localhost
5   tasks:
6     - name: Add a line to a file
7       ansible.builtin.shell:
8         cmd: echo "hello world" >> ./foo.txt
9       changed_when: true
```

# Variables

你可以在 playbooks 中宣告變數，讓後續的任務使用這個變數

使用 `vars` 定義變數之後，你可以使用 Jinja2 filter `"{{ }}"` 將變數轉換成你設定的值

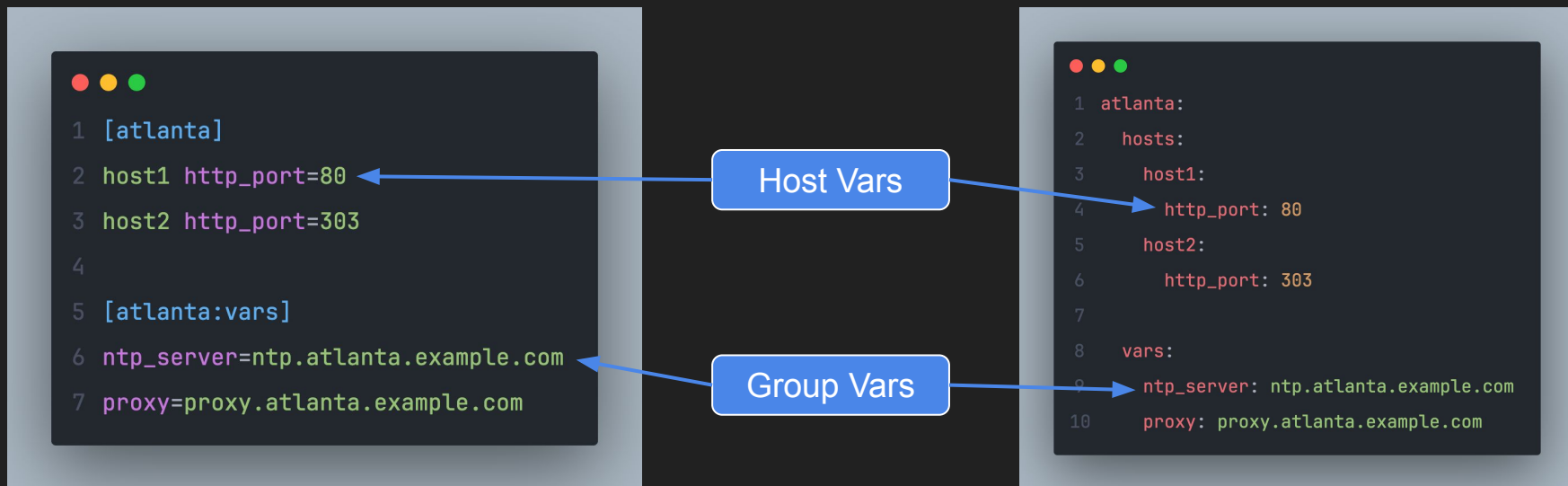
A terminal window with a dark background and light-colored text. It displays an Ansible playbook snippet. The snippet defines a task named 'Show how to use vars' on localhost. It sets a variable 'say\_hello' to 'hello' and uses a debug module to print the value of 'say\_hello' followed by the name 'Howard'. The output of the debug module is shown as 'hello Howard'.

```
1 - name: Show how to use vars
2   hosts: localhost
3   vars:
4     say_hello: 'hello'
5   tasks:
6     - name: Use debug module to show the vars
7       ansible.builtin.debug:
8         msg: "{{ say_hello }} Howard"
```

# 你可以在 Inventory 中設定 Variables

Inventory 中可以設定變數，當中又分為 group vars 與 host vars

Group vars 會作用在所屬底下的全部 host





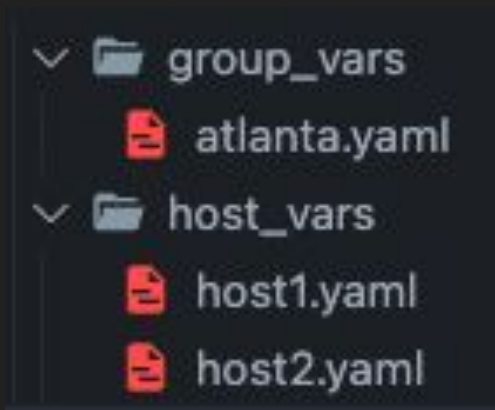
# 如何使用在 Inventory 中的 Variables

使用 Ansible 的 `hostvars` 存取 group vars 與 host vars

```
1 - name: Show how to use vars in inventory
2   hosts: host1
3   tasks:
4     - name: Print the host vars 'http_port'
5       ansible.builtin.debug:
6         msg: "{{ hostvars['host1']['http_port'] }}"
7
8     - name: Print the group vars 'proxy'
9       ansible.builtin.debug:
10        msg: "{{ hostvars['host1']['proxy'] }}"
```

# 使用資料夾整理你的 Variables

可以新增 `group_vars` 與 `host_vars` 資料夾，並在底下建立與 `host` 或 `group` 同名的 YAML 檔案，再把變數寫在裡面



```
1 # host1.yaml
2 http_port: 80
```

# 註冊 Variables

你可以將一個任務的輸出，使用 `register` 註冊為新的變數

```
1 - name: Show how to use register
2   hosts: localhost
3   tasks:
4     - name: Register a output of a command
5       ansible.builtin.command:
6         cmd: "echo 'Hello Tavia!'"
7       changed_when: false
8       register: result
9
10    - name: Print the registered variable
11      ansible.builtin.debug:
12        msg: "{{ result.stdout }}"
13
```

# Conditional

你可以在 playbooks 中使用程式碼常見的控制流程

when      loop

# when

當給定的條件式成立，就執行該任務

```
1 - name: Show how to use when
2   hosts: proxy
3   tasks:
4     - name: Install Nginx on Debian
5       ansible.builtin.apt:
6         name: nginx
7         state: present
8       when: ansible_os_family == "Debian"
9
10    - name: Install Nginx on RedHat
11      ansible.builtin.yum:
12        name: nginx
13        state: present
14      when: ansible_os_family == "RedHat"
```

# Loop

帶入陣列中的值，並重複  
執行任務

```
1 - name: Show how to use loop
2   hosts: localhost
3   tasks:
4     - name: Create users
5       loop:
6         - Johnson
7         - Howard
8         - Elizabeth
9         - Peggy
10        - Tavia
11      ansible.builtin.user:
12        name: "{{ item }}"
13        state: present
```

```
1 - name: Show how to use loop
2   hosts: localhost
3   tasks:
4     - name: Create users
5       loop:
6         - name: Johnson
7           uid: 1001
8         - name: Howard
9           uid: 1002
10        - name: Elizabeth
11          uid: 1003
12      ansible.builtin.user:
13        name: "{{ item.name }}"
14        uid: "{{ item.uid }}"
15        state: present
```

# Role

你可以將你的任務模組化，達到重複利用的效果，類似於 Terraform 的 module

假設你有一個安裝 MySQL 的 playbooks，你可以將其包裝成一個 role。當你之後需要在其他機器安裝 MySQL 時，你可以直接使用這個 role，不需要再寫一次



```
1 # create a new role 'say_hello'  
2 ansible-galaxy init say_hello
```

# Role 的檔案結構

你可以在將 role 放在與 playbooks 檔案同目錄的 `roles` 資料夾底下

也可以將 role 放置在 `/etc/ansible/roles` 資料夾底下



```
1 ---
2 # roles/say_hello/tasks/main.yaml
3 # tasks file for say_hello
4 - name: Say hello
5   ansible.builtin.debug:
6     msg: "Hello!"
7
```

```
1 # playbooks.yaml
2 - name: Show how to use role
3   hosts: localhost
4   roles:
5     - say_hello
```

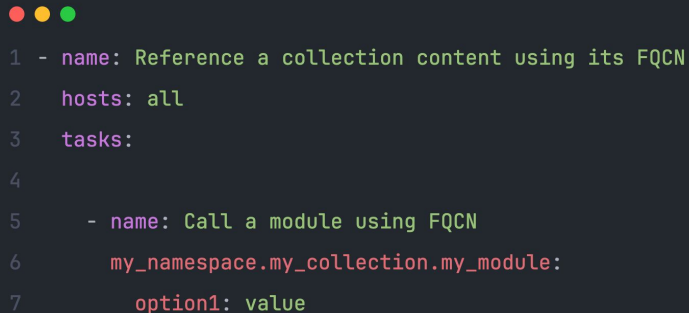


# Collections

也是一種重複使用程式碼的方式

Collections 中會包含 playbooks、roles、modules 與 plugin

與 role 只有任務不同, collection 會描述一個更完整的 IT 基礎架構



```
1 - name: Reference a collection content using its FQCN
2   hosts: all
3   tasks:
4
5     - name: Call a module using FQCN
6       my_namespace.my_collection.my_module:
7         option1: value
```

# Ansible Galaxy

在開始寫新專案前，你可以先在 Ansible Galaxy 上找找有沒有你需要的 role 與 collection

你也可以將你的 role 與 collection 分享到網路上讓別人使用

The screenshot displays the Ansible Galaxy web interface. At the top, there's a navigation bar with a menu icon, the 'A GALAXY' logo, and links for 'About', 'Help', 'Documentation', and 'Login'. Below this is a search bar with the text 'mysql' entered, showing '(1673 results)'. A sidebar on the left contains icons for 'Collections' (65), 'Roles', and 'Plugins'. The main content area lists search results. The first result is the 'mysql' collection by 'community', which has a 4.8/5 score, 10185832 downloads, and is version 3.7.2. It lists 7 modules (mysql\_db, mysql\_info, mysql\_query, etc.), 0 roles, and 2 plugins (mysql, mysql). The second result is 'wp\_mysql\_on\_top\_of\_k8s\_cluster' by 'mohit\_jangir', with 25 downloads and version 1.0.0. A 'Popular Tags' sidebar on the right lists tags like 'system', 'development', 'web', 'monitoring', 'networking', 'database', 'docker', 'cloud', 'security', and 'ubuntu' with their respective download counts.

Tag	Count
system	8,036
development	3,555
web	2,988
monitoring	1,803
networking	1,440
database	1,328
docker	1,317
cloud	1,210
security	1,102
ubuntu	1,085

# 遵循 Ansible 的 Best Practice

[官方文件](#)有列出許多 Ansible 的使用建議，非常推薦一讀

## Best Practices

Here are some tips for making the most of Ansible and Ansible playbooks.

You can find some example playbooks illustrating these best practices in our [ansible-examples repository](#).  
(NOTE: These may not use all of the features in the latest release, but are still an excellent reference!).

# Demo

讓我們用 Ansible playbooks 建立一個 K3s Cluster