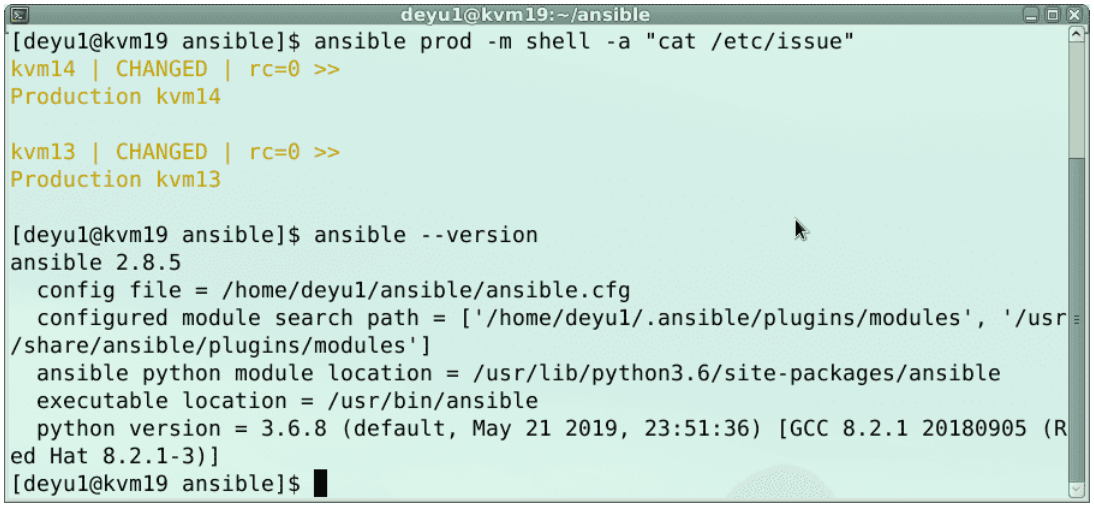
Ansible – Linux 自動化部署

Department of Computer Science and Information Engineering
Chaoyang University of Technology
Taichung, Taiwan, Republic of China

Instructor: De-Yu Wang (王德譽)
E-mail: dywang@mail.cyut.edu.tw
Phone: (04)23323000 ext 4538
Office: E738



Thursday 18th April, 2024

- Instructor: De-Yu Wang
    1. Email: dywang@csie.cyut.edu.tw
    2. Homepage: http://dywang.csie.cyut.edu.tw
    3. 本文件 pdf 檔下載
    4. Phone: (04)23323000 ext 4538
    5. Office: E738
- 參考資料
    1. RedHat Documentation
    2. Ansible User Guide
    3. Ansible 中文權威指南
    4. 現代 IT 人一定要知道的 Ansible 自動化組態技巧

Contents

1 實機練習系統 **1**

1.1 動機 1
1.2 系統設計 1
1.3 練習系統製作-VirtualBox 2
1.4 練習系統製作-Raw Image 5
1.5 NAT 設計 6
1.6 虛擬機管理界面 7
1.7 虛擬機管理界面 V2 8
1.8 系統管理與評分 9
1.9 系統註冊–上課前先確定三件事 11
1.10 虛擬機設定 12
1.11 系統伺服器管理 13

2 Ansible 簡介 **16**

2.1 前言 16
2.2 環境介紹 17
2.3 安裝 17
2.4 * 設定 SSH Keys 18
2.5 * 設定 SUDOER 20
2.6 實機練習 20

3 Inventory 檔案 **21**

3.1 前言 21
3.2 基本設定 21
3.3 變數說明 23
3.4 進階設定 24
3.5 練習系統 inventory 25
3.6 檢查 inventory 26
3.7 實機練習 27

4 組態設定 **28**
4.1 前言 28
4.2 ansible.cfg 範例 28
4.3 練習系統 ansible.cfg 29
4.4 測試 ansible.cfg 30
4.5 實機練習 30

5 Ad-hoc 命令 **32**
5.1 前言 32
5.2 *Shell 命令 32
5.3 模組文件 34
5.4 檔案傳輸 - copy 模組 35
5.5 檔案管理 - file 模組 37
5.6 套件庫 - yum_repository 模組 38
5.7 套件管理 - yum 模組 40
5.8 實機練習 41

6 Playbooks **43**
6.1 前言 43
6.2 第一支 playbook 43
6.3 基本語法 45
6.4 使用變數 47
6.5 loop 迴圈 48
6.6 block 區塊 50
6.7 更新套件 51
6.8 實機練習 52

7 Jinja2 樣板 **54**
7.1 前言 54
7.2 基本語法 54
7.3 變數 55
7.4 debug 輸出 magic 變數 56
7.5 setup 輸出 facts 變數 59
7.6 產生 hosts 檔案 60
7.7 測試 hosts.yml 61
7.8 實機練習 62

8 Roles **63**
8.1 前言 63
8.2 項目結構 63
8.3 初始化角色 apache 64
8.4 編輯角色 apache 任務 65
8.5 編輯角色 apache 樣板 66
8.6 使用角色 apache 68
8.7 實機練習 70

9 RHEL Roles **72**
9.1 前言 72
9.2 安裝 72
9.3 RHEL 角色目錄 73
9.4 使用 RHEL 角色 74
9.5 實機練習 79

10 Ansible Galaxy **80**
10.1 前言 80
10.2 ansible-galaxy 命令 80
10.3 search 81
10.4 install/remove 82
10.5 requirements.yml 83
10.6 requirements.yml 安裝實例 84
10.7 使用 haproxy 角色 85
10.8 使用 phpinfo 角色 87
10.9 實機練習 90

11 LVM 檔案系統 **91**
11.1 LVM 相關模組 91
11.2 *parted 模組 92
11.3 *lvg 模組 94
11.4 lvol 模組 95
11.5 filesystem 模組 96
11.6 移除 lv 97
11.7 實機練習 98

12 Block 錯誤處理 **100**
12.1 Block-rescue-always 100

12.2 變數查詢 101
12.3 實例–正常任務 102
12.4 實例–rescue 104
12.5 實例–debug 105
12.6 實機練習 106

13 Shell 模組 **107**
13.1 前言 107
13.2 模組文件 107
13.3 修改檔案–單一群組 109
13.4 修改檔案–不同群組不同內容 110
13.5 實機練習 112

14 Group 模組 **114**
14.1 前言 114
14.2 模組文件 114
14.3 建立群組帳號 114
14.4 實機練習 115

15 File 模組 **117**
15.1 前言 117
15.2 模組文件 117
15.3 建立目錄 120
15.4 建立軟連結 121
15.5 新增含內容的檔案 122
15.6 實機練習 124

16 Sefcontext 模組 **126**
16.1 前言 126
16.2 模組文件 126
16.3 修改 setype 127
16.4 實機練習 128

17 Lineinfile 模組 **130**
17.1 前言 130
17.2 模組文件 130
17.3 Facts 變數查詢 132
17.4 修改檔案內容 134
17.5 變數不存在 136

17.6 實機練習 138

18 Cron 模組 **140**
18.1 前言 140
18.2 模組文件 140
18.3 建立例行性命令 142
18.4 實機練習 143

19 Vault 加密 **144**
19.1 前言 144
19.2 ansible-vault 命令 144
19.3 建立加密檔案 145
19.4 加解密現有檔案 146
19.5 密碼檔解密 Vault 147
19.6 rekey 重設密碼 148
19.7 實機練習 149

20 User 模組 **150**
20.1 前言 150
20.2 模組文件 150
20.3 建立用户 152
20.4 設定帳號密碼 153
20.5 Vault 密碼檔 154
20.6 建立不同帳號 156
20.7 測試帳號 159
20.8 實機練習 160

Chapter 1

實機練習系統

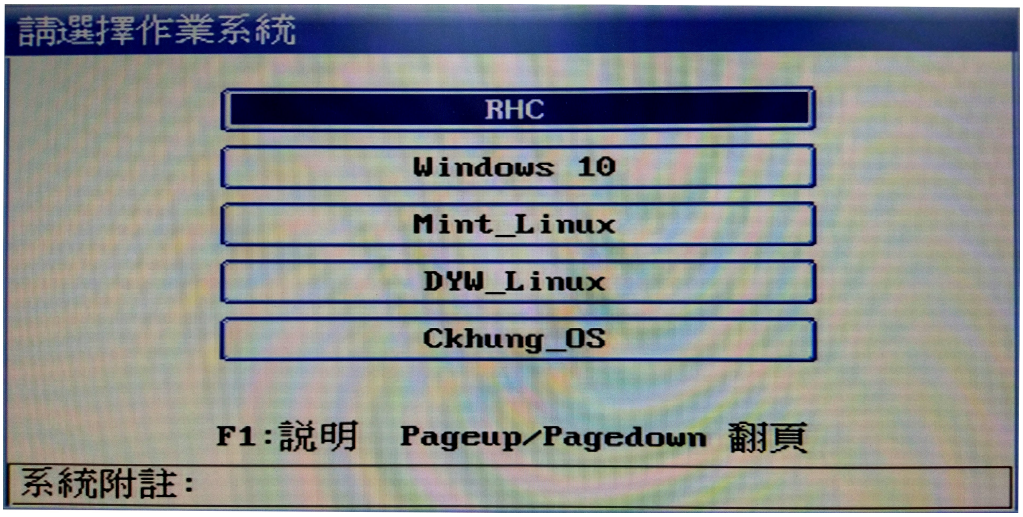
1.1 動機

1. 教學若有實際環境，讓學生實機操作練習，可以大大提昇學習成效。
2. Linux 實機練習，很容易因學生操作錯誤，而造成系統損壞。例如：硬碟分割、格式化及管理的練習。
3. 實機練習結束或操作不當造成當機，系統必須可以迅速還原，才能重複練習。
4. 系統必須可以批次快速修改或更新。
5. 系統必須可以依據學生實際操作狀況，批次進行檢查或評分。
6. 虛擬機練習所需伺服器資源，例如：LDAP 網路帳號、YUM repository 套件庫伺服器、NFS 分享、NTP 校時伺服器、ISCSI 硬碟分享、內部網域 DNS 伺服器、FTP 資源下載... 等環境必須架設好，才能提供虛擬機做各種狀況的練習。

1.2 系統設計

1. Linux 教學系統設計
   (a) 客製化 Linux 系統。
   (b) Linux 伺服器架設。
   (c) VNC 遠端桌面與 KVM 虛擬機。
2. Linux 教學系統安裝

(a) E517 電腦教室：方便學生上課使用。



(b) 個人隨身碟：插上隨身碟，以安裝光碟開機，不需要任何設定，自動安裝於隨身碟，不會影響電腦中的硬碟。

(c) LiveCD：以 LiveCD 光碟開機，開機後即可使用，只是所有操作與設定都是暫存，重新開機後即回到原始狀態。

1.3 練習系統製作-VirtualBox

1. 下載 DYW Linux 練習系統 vdi 檔，下載網址公告於本平台課程內，請登入查看。

```
[root@dywmsi ~]# unxz crt9.vdi.xz
```

2. 依自己使用的 OS (Mac Windows Linux) 下載安裝 VirtualBox。

```
https://www.virtualbox.org/wiki/Downloads
```

3. DYW Linux 練習系統，Host 主機以 virsh 管理 VM，最低硬體因所需 VM 個數不同：

(a) CPU 4 核心以上，若修的課不用內部虛擬機，設定 2 核心也可以。

(b) RAM 8G 以上，若修的課不用內部虛擬機，設定 4G RAM 也可以。

(c) RHCSA, RHCE 請設定 4 核心，8G RAM 以上

(d) CPU 支援虛擬化技術 (virtualization technology)，BIOS 啓用 VT。

4. VirtualBox 新增 VM

```
New(Ctrl+N)
    Name: 自訂，例如 crt9
    Type: Linux
    Version: 只要是 64-bit 就可以，例如 Fedora (64-bit)
    Next >
Memory size
    8192 MB
    Next >
Hard disk
    Use an existing virtual hard disk file
    新增解壓縮的 crt9.vdi，並選用這個硬碟檔。
    Create
```

5. 若要使用內部虛擬機，必須保證巢式虛擬化有開啓，假設匯入的 VM 名稱爲 crt9

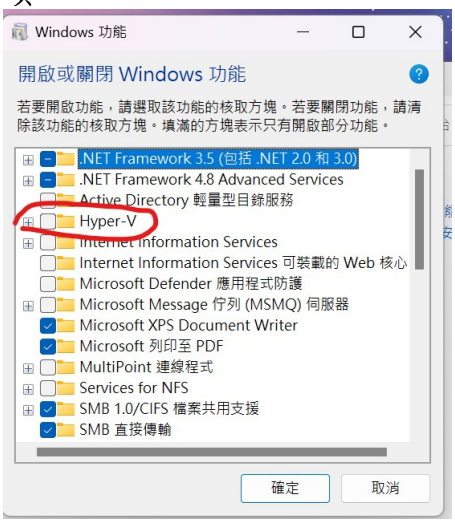
   (a) Host 是 Linux，在終端執行以下命令：

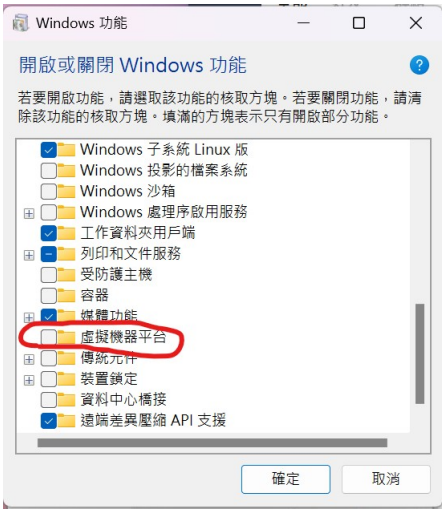
   ```
   [dywang@dywmsi ~]$ VBoxManage modifyvm crt9 --nested-hw-virt on
   ```

   (b) Host 是 window，在命令提示 cmd 執行以下命令：

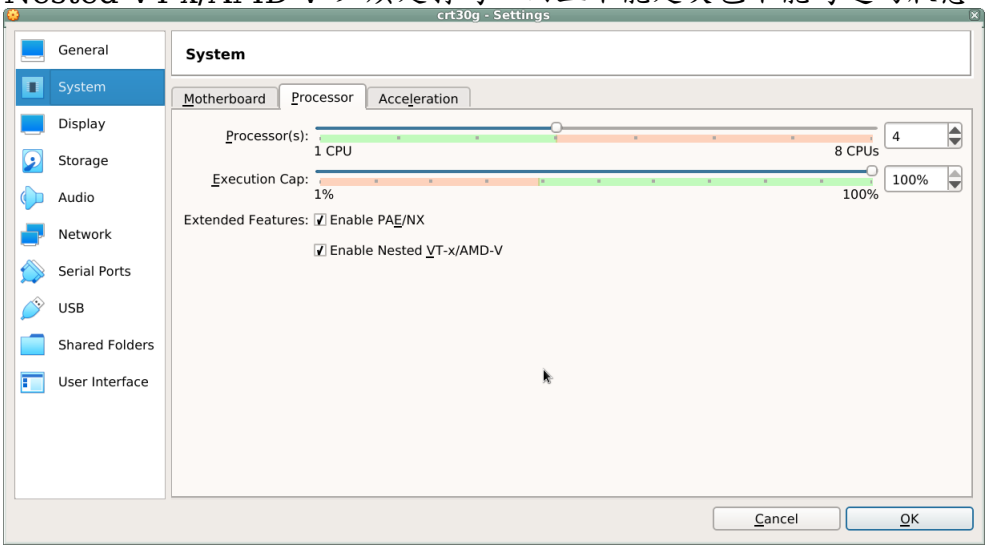
   ```
   cd "c:\Program Files\Oracle\VirtualBox"
   VBoxManage.exe modifyvm crt9 --nested-hw-virt on
   ```

   (c) Host 是 window，還必須關閉「Hyper-V」及「虛擬機器平台」，兩個選項。

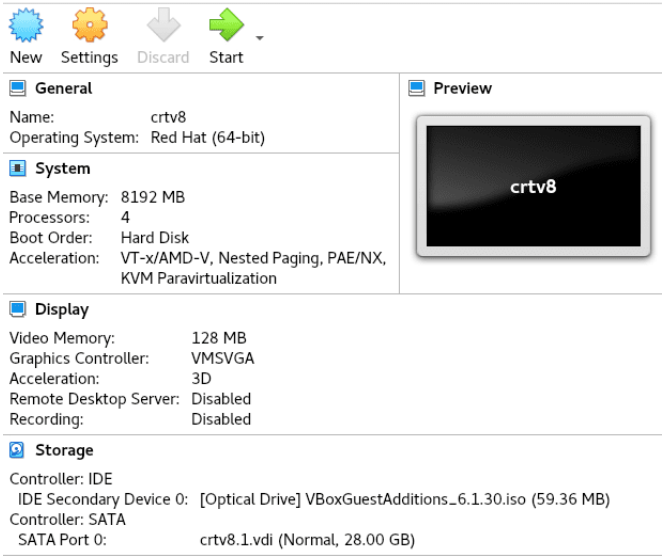




6. 進 VirtualBox 查看 VM crt9 的設定，system -> Processor -> Enable Nested VT-x/AMD-V 必須是打勾，而且不能是灰色不能勾選的狀態。



7. 使用 VirtualBox 或其他虛擬化工具，都要提供 VM crt9 足夠的 CPU 及 RAM，且要開啓巢式虛擬化，只要這三個要求達到，應該不會有□題，其他設定不懂可以先用預設值。若進不了圖形界面，可以打開 Display 的 3D 加速看看，VirtualBox VM 設定 (Settings) 參考值如下：



8. * 爲方便 VirtualBox 使用，建議下載附加套件 Oracle_VM_VirtualBox_Extension_Pack-6.1.30.vbox-extpack，放到你的 VirtualBox 目錄中，並於 VirtualBox 選單中加入附加 CD image。

```
Devices --> Insert Guest Additions CD image...
```

9. 開啓 DYW Linux 後，若虛擬機沒安裝，請先 rebuild 安裝虛擬機，練習後只需要 revert 還原，不用再要重新 rebuild，rebuild 要從學校 server 下載 image，要花較多時間。

1.4 練習系統製作-Raw Image

1. 下載 DYW Linux 練習系統 image 檔，或由 VDI 檔轉換格式，下載網址公告於本平台課程內，請登入查看。

2. 下載的 image 檔是 xz 壓縮檔，先用 unxz 解壓縮，windows 可用 winzip 解壓縮。

```
1 [root@dywmsi ~]# unxz crt9.img.xz
```

3. DYW Linux 練習系統，Host 主機以 virsh 管理 VM，最低硬體因所需 VM 個數不同：

   (a) CPU 4 核心以上，若修的課不用內部虛擬機，設定 2 核心也可以。

   (b) RAM 8G 以上，若修的課不用內部虛擬機，設定 4G RAM 也可以。

(c) RHCSA, RHCE 請設定 4 核心，8G RAM 以上

(d) CPU 支援虛擬化技術 (virtualization technology)，BIOS 啓用 VT。

4. 將 image 檔 dd 寫到 32G 以上硬碟或隨身碟，其中 /dev/sdX 是您要寫入的硬碟或隨身碟。

```
1 [root@dywmsi ~]# dd if=crt9.img of=/dev/sdX bs=100M
```

5. /dev/sdX 整顆硬碟的資料會被覆蓋，請確認要寫入的硬碟無誤再按 Enter 執行。

6. dd 完成後以 /dev/sdX 開機。

7. 開啓 DYW Linux 後，若虛擬機沒安裝，請先 rebuild 安裝虛擬機，練習後只需要 revert 還原，不用再要重新 rebuild，rebuild 要從學校 server 下載 image，要花較多時間。

1.5 NAT 設計

1. NAT 架設：電腦教室網路設置、管控學生上課上網。

2. 在 Linux，一行命令即可完成設定；在 Windows，Honda 老師亦幫忙撰寫批次檔。控制方式：

3. 任何設定皆不管控老師機，老師機可自由上網。

4. 學生機上網方式：

(a) 自由上網

(b) 不能上網

(c) 限制只能上學內網

(d) 限制只能上某一網站

(e) 限制只能上「依老師設定」的部分網站

(f) 只有某一學生機可自由上網

5. 不管老師做任何網路設定，下課後系統自動恢復成所有電腦皆可自由上網，以避免因老師忘了復原，而影響下節上課的老師。

1.6 虛擬機管理界面

1. 設計虛擬機管理 GUI 程式，方便學生管理虛擬機。程式包含虛擬機重安裝、還原、重新啓動及終端界面等四個功能。

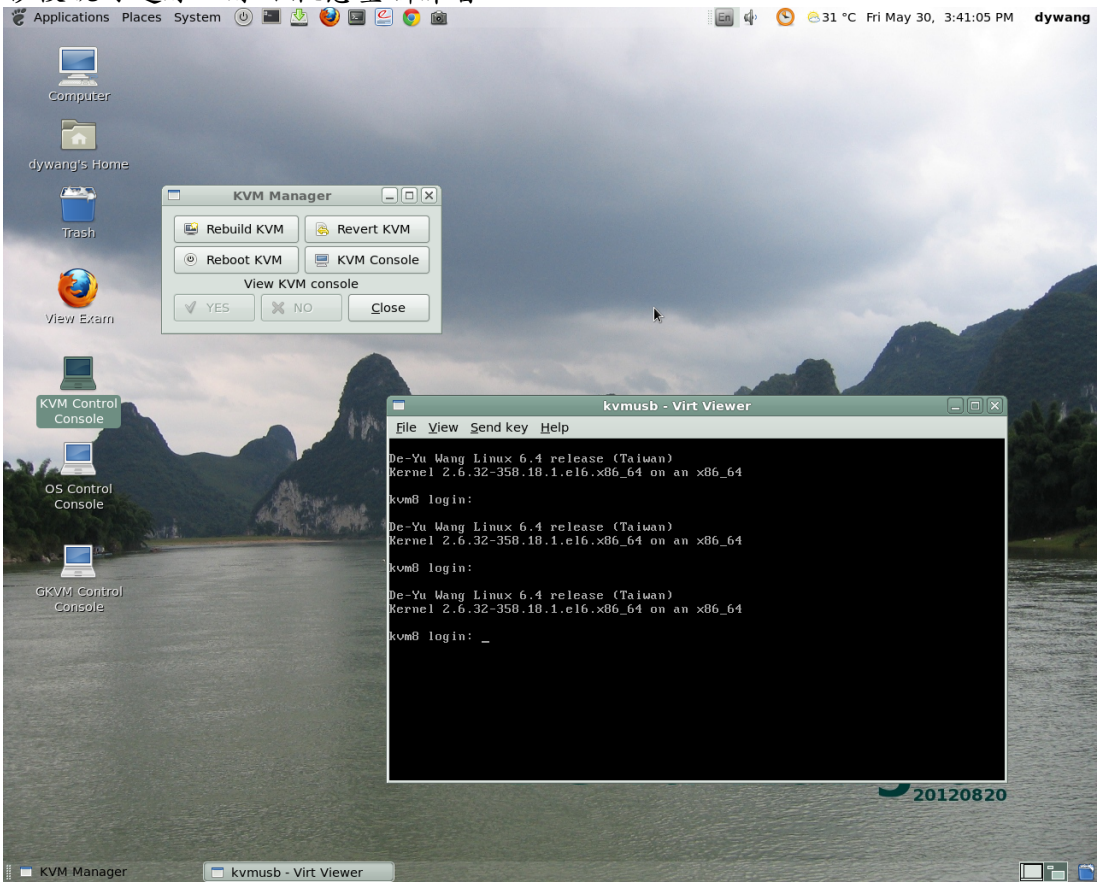
   (a) Rebuild KVM：重新安裝 KVM 虛擬機，約 10 幾分鐘。

   (b) Revert KVM：還原 KVM 虛擬機，約幾秒鐘到幾分鐘，請耐心等侯，不要在還原過程按管理界面的其他功能，以免造成虛擬機損壞。

   (c) Reboot KVM：重新啓動 KVM 虛擬機。

   (d) KVM Console：打開 KVM 虛擬機終端畫面。

2. 系統桌面中黑色背景視窗爲 KVM 虛擬機操作界面。原始 KVM 虛擬機主機名稱爲 111111，因已練習過，主機也已被設定成 kvm8，點擊 Revert KVM，幾秒後就可還原回原始狀態重新練習。



3. 在終端機以指令 virt-view 直接開啓虛擬機，出現錯誤訊息。

```
[dywang@dywH ~]$ sudo virt-viewer kvmusb
No protocol specified
Cannot open display:
Run 'virt-viewer --help' to see a full list of available command line
    options
```

4. 先檢查 Xwindow 認證變數 $XAUTHORITY 存不存在？

```
[dywang@dywH ~]$ echo $XAUTHORITY
```

5. 變數 XAUTHORITY 不存在，必須先設定 XAUTHORITY=~/.Xauthority 後再使用 sudo 啓動虛擬機，才能正常啓動。

```
[dywang@dywH ~]$ XAUTHORITY=~/.Xauthority sudo virt-viewer kvmusb
```

1.7 虛擬機管理界面 V2

1. 設計虛擬機管理 GUI 程式，方便學生管理虛擬機。程式包含虛擬機重安裝、還原、重新啓動、終端界面、啓用實體 USB 隨身碟、解除實體 USB 隨身碟的使用等六個功能。



(a) Rebuild KVM：重新安裝 KVM 虛擬機，直接下載 qcow2 映像檔，產生虛擬機，校內網路約 1 分多鐘。

(b) Revert KVM：還原 KVM 虛擬機，約幾秒鐘到幾分鐘，請耐心等侯，務必等黑色終端出現登入畫面，表示還原完成。練習前先關閉黑色終端畫面讓還原程式拍照，記下還原狀態，以免造成虛擬機損壞。

(c) Reboot KVM：重新啓動 KVM 虛擬機。

(d) KVM Console：打開 KVM 虛擬機終端畫面。

(e) Attach USB：插上 USB 隨身碟，按下此按鈕，KVM 虛擬機啓動實體 USB 隨身碟。

(f) Detach USB：KVM 虛擬機解除實體 USB 隨身碟的使用。

1.8 系統管理與評分

1. 系統管理腳本程式自動掃瞄目前開機的電腦，選擇要進行的工作，輸入密碼後即可自動完成。

```
[root@dywH ~]# e517sshpass.sh
myip=104
hosts=103,105
q(quit) iscsi halt reboot c(clear) l(lvm) rhcsa rhce
>  t(mounttest) k(kvmreboot) v(kvmrevert) s(servicerestart)
>  vi1 c1 lpvar c999 sh999 hp(hostpatch)? c999
keyin remote host root password:
```

2. 預設工作項目：

   (a) q(quit)：退出，無動作。

   (b) iscsi：學生機 ISCSI 硬碟分享架設。

   (c) halt：學生機關機。

   (d) reboot：學生機重新啓動。

   (e) c(clear)：登入學生機後清除部分資料，此部分保留修改彈性，目前入作爲限制學生機互相登入之限制，以防代作答情況。

   (f) l(lvm)：硬碟分割、格式化、自動掛載評分。

   (g) rhcsa：RHCSA 考照練習總評分。

   (h) rhce：RHCE 考照練習總評分。

   (i) t(mounttest)：測試學生機是否有掛載隨身碟或光碟。

   (j) k(kvmreboot)：學生機 KVM 虛擬機重新啓動。

   (k) v(kvmrevert)：學生機 KVM 虛擬機還原。

   (l) s(servicerestart)：啓動學生機主機服務。

   (m) vi1：vi 編輯器實機練習評分。

   (n) c1：C 語言第一支程式練習評分。

   (o) lpvar：shell 變數變化練習評分。

   (p) c999：C 語言程式實機練習評分。

   (q) sh999：Shell script 程式練習評分。

   (r) hp(hostpatch)：學生機網路修正後重新啓動網路。

3. 遠距教學，在家練習，自己評分上傳：

(a) 登入本平台，下載評分腳本及參數檔，以「資料庫」第一次評分爲例。

(b) 切換身分爲 root，root 密碼在平台部落格，請登入。

```
[dywang@deyu ~]$ su -
Password:
[root@deyu ~]#
```

(c) 切換工作目錄到評分程式的下載目錄，用瀏覽器下載預設是放在家目錄的 Downloads。

```
[root@deyu ~]# cd /home/dywang/Downloads
```

(d) 下載的評分程式沒有可執行的權限，先增加評分程式 mariadbeval1.sh.x 可執行屬性，使評分腳本 mariadbeval1.sh.x 成爲可執行檔。

```
[root@deyu Downloads]# chmod +x mariadbeval1.sh.x
```

(e) 查看目前有兩個檔案，評分程式爲 mariadbeval1.sh.x，參數檔爲 mariadbpara1.sh。

```
[root@deyu Downloads]# ll
-rwxrwxr-x. 1 root root 58656 May 21 13:29 mariadbeval1.sh.x
```

(f) 執行評分腳本 mariadbeval1.sh.x

```
[root@deyu Downloads]# ./mariadbeval1.sh.x
20210521-13:46:36
./mariadbeval1.sh.x
163.17.10.123
kvm=kvm19
163.17.10.123 10927000 王大呆
mariadbpara1.sh
mariadbeval1.sh.x_kvm19_1.mysqlversion:0
mariadbeval1.sh.x_kvm19_2.mariadb_enabled:0
mariadbeval1.sh.x_kvm19_3.mariadb_active:0
mariadbeval1.sh.x_kvm19_4.rootpw:0
```

(g) 如果評分參數檔不在同目錄，評分腳本 mariadbeval1.sh.x 無法執行。

```
[root@deyu Downloads]# mv mariadbpara1.sh /tmp/
```

```
[root@deyu Downloads]# ./mariadbeval1.sh
20210521-13:48:51
./mariadbeval1.sh
163.17.10.123
kvm=kvm19
163.17.10.123 10927000 王大呆
ls: cannot access mariadbpara1: No such file or directory
NO mariadbpara1 file
```

1.9 系統註冊-上課前先確定三件事

1. E517 電腦教室

   (a) E517 教室位置，靠窗爲第一排，確認自己電腦位置爲?-?。例如：1-1。

   (b) 在伺服器 (也就是開機進入圖形界面中有虛擬機管理界面的這台主機)，不是虛擬機，確認自己的電腦 IP。以下例子爲 192.168.1.140，IP 尾數爲 140。

```
[dywang@deyu ~]$ ifconfig | grep 192
          inet addr:192.168.1.140  Bcast:192.168.1.255  Mask
              :255.255.255.0
          inet addr:192.168.122.1  Bcast:192.168.122.255  Mask
              :255.255.255.0
```

   (c) 在伺服器 deyu.wang 主機一般帳號 dywang 家目錄，產生學號姓名檔 sid，檔案資訊格式爲「電腦位置 IP 尾數學號姓名」，每一欄位中間以一個空格隔開。以下例子電腦位置 1-11、IP 尾數 199、學號 123456，姓名王大呆。

```
[dywang@deyu ~]$ echo '1-11 199 123456 王大呆' > ~/sid
[dywang@deyu ~]$ cat ~/sid
1-11 199 123456 王大呆
```

2. 在家練習

   (a) 查詢練習系統對外 public ip 有三種方式：

```
[dywang@deyu ~]$ curl ifconfig.me
163.17.10.122[dywang@deyu ~]$
[dywang@deyu ~]$ curl ipinfo.io/ip
163.17.10.122[dywang@deyu ~]$
```

```
[dywang@deyu ~]$ curl icanhazip.com
163.17.10.122
```

(b) 將 ip 學號姓名寫入 dywang 家目錄的 sid 檔，如下範例：

```
[dywang@deyu ~]$ echo '163.17.10.123 10827000 王大呆' > sid
```

(c) 查看 sid，ip 學號姓名務必正確，以免評分錯誤或無法匯入成績。

```
[dywang@deyu ~]$ cat sid
163.17.10.123 10827000 王大呆
```

1.10 虛擬機設定

1. 伺服器(也就是開機進入圖形界面中有虛擬機管理界面的這台主機)，不確定的話，先看命令列提示符號是不是 [dywang@deyu ~]$，或是 ifconfig 查目前所在主機的 IP：

   (a) Server 主機有兩個名稱 deyu.wang 及 server.deyu.wang

   (b) Server 與虛擬機網域 deyu.wang，使用網段 192.168.122.0/24。

   (c) Server IP 爲 192.168.122.1，虛擬機 IP 爲 192.168.122.X ，其中 X 依題目要求設定。

2. 虛擬機環境說明：

   (a) 虛擬機主機名稱 kvmX.deyu.wang IP 192.168.122.X，X 爲虛擬機編號，例如 5 或 7。

   (b) 虛擬機 root 密碼無法取得，統一重新設定爲 123qwe。

   (c) RHCE 練習環境增加
   網域 my111.wang，使用網段 192.168.111.0/24，
   server 主機增加此網域 IP 192.168.111.1。

   (d) 虛擬機網路設定：

```
IP Address:   192.168.122.X
Netmask:      255.255.255.0
Gateway:      192.168.122.1
Name server:  192.168.122.1
```

(e) RHCSA/RHCE 6,7 repository 套件庫網址：
http://dywang.csie.cyut.edu.tw/centos7.

(f) RHCSA 8 repository 套件庫網址：

i. BaseOS repo server:
http://dywang.csie.cyut.edu.tw/centos8/BaseOS.

ii. AppStream repo server:
http://dywang.csie.cyut.edu.tw/centos8/AppStream.

(g) kernel packages: 網路下載核心 RPM 檔
http://dywang.csie.cyut.edu.tw/kernel/centos7.

(h) LDAP server 設定：

i. 系統 deyu.wang

ii. base DN 爲 dc=deyu,dc=wang

iii. 證書 ftp://deyu.wang/pub/cacert.pem.

1.11 系統伺服器管理

1. 使用虛擬機練習時，伺服器 (也就是開機進入圖形界面中有虛擬機管理界面的這台主機) 必須正常運作，否則有些驗證無法進行。

2. 練習用到的 server 都已寫成腳本，如果運作有問題，可以執行腳本重啓。

1.11.1 DNS server

(a) 如果確定你的練習虛擬機 DNS 設定沒問題，但 ssh 卻無法使用主機名稱連線到虛擬機，可以在 server 機使用 host 命令看是否可以查到虛擬機的 IP？

(b) 沒有 DNS server 可查。

```
[dywang@deyu ~]$ host kvm7.deyu.wang
;; connection timed out; trying next origin
;; connection timed out; no servers could be reached
```

(c) 在 DNS server 查不到 kvm7.deyu.wang 的 IP。

```
[dywang@deyu ~]$ host kvm7.deyu.wang
Host kvm7.deyu.wang not found: 3(NXDOMAIN)
```

(d) 在伺服器 deyu.wang 主機切換成 root 身份。

```
[dywang@deyu ~]$ su -
Password: 123123
[root@deyu ~]#
```

(e) 執行 setdnsmasq.sh 腳本重啓 dnsmasq DNS 服務。

```
[root@deyu ~]# setdnsmasq.sh restart
```

(f) 再使用 host 查詢 kvm7.deyu.wang，已可查到其 IP 爲 192.168.122.7。

```
[root@deyu ~]# host kvm7.deyu.wang
kvm7.deyu.wang has address 192.168.122.7
kvm7.deyu.wang mail is handled by 1 20.
```

(g) 如果執行 setdnsmasq.sh 腳本重啓 dnsmasq DNS 服務未能解決問題，還可以使用 setdns.sh 腳本啓動 named DNS 服務。

```
[root@deyu ~]# setdns.sh restart
```

1.11.2 LDAP server

(a) 如果確認你的虛擬機 LDAP 設定沒問題，但還是無法以 ldapuser1 登入。在伺服器 deyu.wang 主機 (不是你的練習虛擬機)，先切換成 root 身份。

```
[dywang@deyu ~]$ su -
Password: 123123
[root@deyu ~]#
```

(b) 在伺服器執行 setldap1.sh restart 重新產生 LDAP 憑證。因爲憑證是新產生的，所以虛擬機必須再重新執行認證設定更新。

```
[root@deyu ~]# setldap1.sh restart
```

1.11.3 NFS server

(a) 虛擬機設定自動掛載必須查得到 server 有分享目錄，server 的 NFS 不像 LDAP 有憑證過期問題，所以一般都不需要重啓，但如果要重啓 NFS server，先在伺服器 deyu.wang 主機切換成 root 身份。

```
[dywang@deyu ~]$ su -
Password: 123123
[root@deyu ~]#
```

(b) 執行 setnfs.sh 腳本重新啓動。

```
[root@deyu ~]# setnfs.sh restart
```

Chapter 2

Ansible 簡介

2.1 前言

1. Ansible 是一種開源配置工具，Linux 系統管理員使用的自動化工具。
2. Ansible 能讓系統管理員從一個控制節點（即 Ansible 服務器）管理數百台服務器。
3. Ansible 不需要任何代理，並且可以在 SSH 和 python 上工作。
4. 觀念說明：
   (a) 控制節點 (Control node)：安裝 Ansible 的主機，可以運行 ansible 相關命令。
   (b) 受管節點 (Managed nodes)：被 Ansible 管理的主機。
   (c) Inventory：受管節點的列表，inventory 檔案有時也稱爲 hostfile。
   (d) 模組 (Modules)：Ansible 執行單元，種類有套件 (Packaging)、資料庫 (Database)、通知 (Notification)、雲端 (Cloud)、Windows 到網路 (Network) 等。要是找不到想要的功能，還可以透過 command 和 shell 等 Modules 自行擴充。
   (e) 任務 (Tasks)：Ansible 中的工作單位，可以使用 ad-hoc 命令一次執行一個任務。
   (f) Playbooks：Ansible 配置、部署等任務的腳本，腳本描述希望遠端主機執行命令的方案，或者一組 IT 程式執行的命令集合，Playbooks 可以包含變數。

2.2 環境介紹

1. 實作環境：CentOS 8
2. 練習系統：
   (a) 控制節點 (Control node) ：kvm19.deyu.wang 192.168.122.19
   (b) 節點一 (Managed node 1) ：kvm11.deyu.wang 192.168.122.11
   (c) 節點二 (Managed node 2) ：kvm12.deyu.wang 192.168.122.12
   (d) 節點三 (Managed node 3) ：kvm13.deyu.wang 192.168.122.13
   (e) 節點四 (Managed node 4) ：kvm14.deyu.wang 192.168.122.14
   (f) 節點五 (Managed node 5) ：kvm15.deyu.wang 192.168.122.15
3. 所有系統的 root 密碼是 123qwe。
4. 若需要安裝套件，套件庫位於 dywang.csie.cyut.edu.tw：
   http://dywang.csie.cyut.edu.tw/centos8/BaseOS
   http://dywang.csie.cyut.edu.tw/centos8/AppStream

2.3 安裝

1. Ansible 已有針對 RHEL/CentOS 8 打包好的 rpm，可下載安裝，本練習已將 ansible 相關套件存放於 dywang server，只要設定好 repository 即可使用。

```
[root@kvm19 ~]# cat /etc/yum.repos.d/ansible.repo
[Ansible]
name=Ansible
baseurl=http://dywang.csie.cyut.edu.tw/materials/ansible
gpgcheck=0
```

2. yum 安裝 ansible。

```
[root@kvm19 ~]# yum install ansible
```

3. ansible 開頭的命令。

```
[root@kvm19 ~]# ansible[TAB][TAB]
ansible             ansible-doc         ansible-pull
ansible-config      ansible-galaxy      ansible-test
```

```
ansible-connection  ansible-inventory   ansible-vault
ansible-console     ansible-playbook
```

4. 查看 ansible 版本 2.9.5，python 版本 3.6.8。

```
[root@kvm19 ~]# ansible --version
ansible 2.9.5
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules',
 '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.6/site-packages/
      ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, May 21 2019, 23:51:36)
 [GCC 8.2.1 20180905 (Red Hat 8.2.1-3)]
```

2.4 * 設定 SSH Keys

1. Ansible 控制節點與受管節點間藉由 ssh 連線，必須設定免密碼連線，或執行 ansible 時使用 -ask-pass 選項要求輸入密碼。爲方便自動完成任務一般會使用免密碼連線。

2. 產生一組金鑰

```
[root@kvm19 ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:F9OoHjIT3jPlN95CnwEAKnrIQsySYaZMQOZU6wVpwsA root@kvm19.deyu.wang
The key's randomart image is:
+---[RSA 2048]----+
|XB.o.    ...     |
|%E oo   .   +    |
|+*o...o   = o    |
|o..o.o o + o .   |
|. +.. = S o + .  |
| . .   = = + + o |
|        .   o +  |
|             .   |
|                 |
+----[SHA256]-----+
```

3. 查看.ssh 目錄下新增了 id_rsa 私鑰及 id_rsa.pub 公鑰。

```
[root@kvm19 ~]# ll .ssh/
total 12
-rw-r--r--. 1 root root  401 Mar  2 10:32 authorized_keys
-rw-------. 1 root root 1831 Mar  5 10:59 id_rsa
-rw-r--r--. 1 root root  402 Mar  5 10:59 id_rsa.pub
```

4. ssh-copy-id 將 id_rsa.pub 公鑰匯入到 kvm11，輸入密碼完成匯入。

```
[root@kvm19 ~]# ssh-copy-id kvm11.deyu.wang
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
 "/root/.ssh/id_rsa.pub"
The authenticity of host 'kvm11.deyu.wang (192.168.122.11)' can't be
    established.
ECDSA key fingerprint is SHA256:l6jIMoK+
    z82ydTlG8MKarNcUDLVSfjCRjWRzhGOtDNc.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s),
 to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed --
 if you are prompted now it is to install the new keys
root@kvm11.deyu.wang's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'kvm11.deyu.wang'"
and check to make sure that only the key(s) you wanted were added.
```

5. ssh 試著連線 kvm11，不用輸入密碼即可登入。

```
[root@kvm19 ~]# ssh kvm11.deyu.wang
Last login: Mon Mar  2 18:40:01 2020 from 192.168.122.1
```

6. 退出 kvm11。

```
[root@kvm11 ~]# exit
logout
Connection to kvm11.deyu.wang closed.
```

2.5 * 設定 SUDOER

1. 控制節點以一般用户 deyu1 執行 ansible 時，受管節點也是以用户 deyu1 權限執行，必須以 root 權限才能執行的命令要用 sudo 執行。

2. 本練習系統 ansible 控制節點及受管節點都已建立 deyu1 帳號，將此帳號設定成免密碼可以使用 sudo 執行所有命令。

```
[root@kvm19 ~]# vim /etc/sudoers.d/ansible
[root@kvm19 ~]# cat /etc/sudoers.d/ansible
deyu1 ALL=(ALL) NOPASSWD: ALL
```

3. 其他受管節點也做相同的 sudoer 設定。

2.6 實機練習

1. 主機說明：

   (a) 控制節點 (Control node) ：kvm19.deyu.wang 192.168.122.19

   (b) 節點一 (Managed node 1) ：kvm11.deyu.wang 192.168.122.11

   (c) 節點二 (Managed node 2) ：kvm12.deyu.wang 192.168.122.12

   (d) 節點三 (Managed node 3) ：kvm13.deyu.wang 192.168.122.13

   (e) 節點四 (Managed node 4) ：kvm14.deyu.wang 192.168.122.14

   (f) 節點五 (Managed node 5) ：kvm15.deyu.wang 192.168.122.15

2. 在控制主機 kvm19，完成以下工作：

   (a) 安裝 ansible 套件。

   (b) 查詢 ansible 版本，導向到 /tmp/ansibleversion.txt

3. * 使用同一組 ssh keys，讓控制節點與受管節點間，用户 root 彼此都可免密碼登入，用户 deyu1 彼此也都可免密碼登入。

4. * 設定控制主機及所有節點，都可以免密碼執行 sudo。

Chapter 3

Inventory 檔案

3.1 前言

1. Ansible 可透過 inventory 檔案配置，同時操作屬於一個組的多台主機。
2. Inventory 檔案預設爲 /etc/ansible/hosts。
3. 也可以下命令時使用 -i <path> 指定 inventory 檔案，而不使用預設檔。
4. 還可以同時使用多個 inventory 檔案，也可以從動態源或雲端取得 inventory 配置資訊。

3.2 基本設定

1. Ansible 套件檔案 /etc/ansible/hosts，是一個 inventory 範例檔。範例說明：

```
[root@kvm19 ~]# cat /etc/ansible/hosts
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups
```

2. 範例一：不屬於群組的主機。

```
# Ex 1: Ungrouped hosts, specify before any group headers.

```

```
## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10
```

3. 範例二：屬於 webservers 群組的主機。

```
# Ex 2: A collection of hosts belonging to the 'webservers' group

## [webservers]
## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110
```

4. 使用 pattern 指定多台主機，以下範例是 www1.example.com 至 www6.example.com 共 6 台主機。

```
# If you have multiple hosts following a pattern you can specify
# them like this:

## www[001:006].example.com
```

5. 範例二：屬於 dbservers 群組的主機。

```
# Ex 3: A collection of database servers in the 'dbservers' group

## [dbservers]
##
## db01.intranet.mydomain.net
## db02.intranet.mydomain.net
## 10.25.1.56
## 10.25.1.57
```

6. 另一種 pattern (最前面的 0 省略) 指定多台主機，以下範例 db-99-node.example.com 至 db-101-node.example.com 共 3 台主機。

```
# Here's another example of host ranges, this time there are no
# leading 0s:

## db-[99:101]-node.example.com
```

7. pattern 除了數字，也可以定義字母範圍，以下範例 db-a.example.com 至 db-f.example.com 共 6 台主機。

```
db-[a:f].example.com
```

3.3 變數說明

1. 連線的遠端主機名稱，例如前一節設定別名時使用。

```
1 ansible_ssh_host
```

2. ssh 埠號，如果不是預設的埠號，可透過此變數設定。

```
1 ansible_ssh_port
```

3. 預設的 ssh 使用者名稱

```
1 ansible_ssh_user
```

4. ssh 密碼，使用變數設定 ssh 密碼不安全，建議使用 –ask-pass 或 SSH 金鑰。

```
1 ansible_ssh_pass
```

5. sudo 密碼，使用變數設定 sudo 密碼不安全，建議使用 –ask-sudo-pass。

```
1 ansible_sudo_pass
```

6. sudo 命令路徑

```
1 ansible_sudo_exe
```

7. 與主機的連線類型，例如:local, ssh 或者 paramiko。

```
ansible_connection
```

8. ssh 使用的私鑰檔案

```
ansible_ssh_private_key_file
```

9. 目標系統的 shell 類型，預設使用 sh 語法

```
ansible_shell_type
```

3.4 進階設定

1. 埠號：如果有主機的 SSH 不是標準的 22 port，可在主機名之後加上埠號，用冒號分隔。

```
www1.example.com:2299
```

2. 別名：node1 會連線 192.168.1.11:2299

```
node1 ansible_ssh_port=2299 ansible_ssh_host=192.168.1.11
```

3. 組的子成員：以下範例 webservers, dbservers 群組都是 servers 群組的子成員。

```
[webservers]
alpha.example.org
beta.example.org
192.168.1.100
192.168.1.110

[dbservers]
db01.intranet.mydomain.net
db02.intranet.mydomain.net
10.25.1.56
10.25.1.57

[servers:children]
```

```
webservers
dbservers
```

3.5 練習系統 inventory

1. 控制節點 kvm19 有一用户 deyu1，以此用户產生 inventory 檔 /home/deyu1/ansible/inventory，要求如下：
   (a) kvm11 是 dev 主機群組的成員
   (b) kvm12 是 test 主機群組的成員
   (c) kvm13 及 kvm14 是 prod 主機群組的成員
   (d) kvm15 是 balancers 主機群組的成員
   (e) prod 群組是 webservers 主機群組的成員
2. 切換成用户 deyu1

```
[root@kvm19 ~]# su - deyu1
```

3. 在家目錄建立 ansible 目錄

```
[deyu1@kvm19 ~]$ mkdir ansible
```

4. 切換到 ansible 目錄

```
[deyu1@kvm19 ~]$ cd ansible
```

5. 在 ansible 目錄編輯檔案 inventory

```
[deyu1@kvm19 ansible]$ vim inventory
```

6. inventory 檔案依要求設定如下，所有受管主機的登入密碼都爲 123qwe：

```
[deyu1@kvm19 ansible]$ cat inventory
[all:vars]
ansible_password=123qwe

[dev]
kvm11

[test]
kvm12

[prod]
kvm13
kvm14

[balancers]
kvm15

[webservers:children]
prod
```

3.6 檢查 inventory

1. 使用 inventory 列出群組 prod 主機：

```
[deyu1@kvm19 ansible]$ ansible prod -i inventory --list-hosts
  hosts (2):
    kvm13
    kvm14
```

2. 使用 /home/deyu1/ansible/inventory 列出所用受管主機，pattern 'all' 也可用'*' 代表全部：

```
[deyu1@kvm19 ansible]$ ansible all -i inventory --list-hosts
  hosts (5):
    kvm11
    kvm12
    kvm15
    kvm13
    kvm14
```

3.7 實機練習

1. 控制節點 kvm19 有一用户 deyu1，以此用户管理 ansible。
2. 在家目錄 /home/deyu1 下建立工作目錄 ansible。
   遠距教學期間，同學在家練習時，工作目錄改爲學號不加 s，例如 10927000，以方便上傳評分。
3. 在 ansible 工作目錄下產生 inventory 檔，要求如下：
   (a) kvm11 是 dev 主機群組的成員
   (b) kvm12 是 test 主機群組的成員
   (c) kvm13 及 kvm14 是 prod 主機群組的成員
   (d) kvm15 是 balancers 主機群組的成員
   (e) prod 群組是 webservers 主機群組的成員
4. 使用 inventory 檔列出所有群組主機。
5. 使用 inventory 檔列出群組 webservers 主機。

Chapter 4

組態設定

4.1 前言

1. Ansible 支援多種來源設定組態 (Configuration)，包含名爲 ansible.cfg 的 ini 、環境變數、命令選項、playbook 關鍵字、變數等。
2. ansible.cfg 讓使用者看到所有可用的組態設定、其預設值、如何設定等。
3. 因有多種組態設定來源，可能不一致，Ansible 以下列順序搜尋，先找到後其他就忽略。
   (a) 環境變數 ANSIBLE_CONFIG
   (b) 現在工作目錄下的 ansible.cfg
   (c) 使用家目錄下的 ~/.ansible.cfg
   (d) 系統 /etc 目錄下的 /etc/ansible/ansible.cfg

4.2 ansible.cfg 範例

1. Ansible 在 /etc/ansible 目錄下有組態設定範例檔 ansible.cfg，包含以下章節：

```
[root@kvm19 ansible]# egrep -v '^(#|$)' /etc/ansible/ansible.cfg
[defaults]
[inventory]
[privilege_escalation]
[paramiko_connection]
[ssh_connection]
[persistent_connection]
[accelerate]
[selinux]
[colors]
```

```
11 [diff]
```

2. 查看 [defaults] 章節的部分設定範例如下，例如：inventory 設定 inventory 檔的位置，role_path 設定使用於 playbooks 的角色的位置。

```
1  [root@kvm19 ansible]# cat /etc/ansible/ansible.cfg | \
   grep '^\[defaults' -A60 | egrep '^(\[|#[a-z])'
3  [defaults]
   #inventory      = /etc/ansible/hosts
5  #library        = /usr/share/my_modules/
   #module_utils   = /usr/share/my_module_utils/
7  #remote_tmp     = ~/.ansible/tmp
   #local_tmp      = ~/.ansible/tmp
9  #plugin_filters_cfg = /etc/ansible/plugin_filters.yml
   #forks          = 5
11 #poll_interval  = 15
   #sudo_user      = root
13 #ask_sudo_pass = True
   #ask_pass      = True
15 #transport      = smart
   #remote_port    = 22
17 #module_lang    = C
   #module_set_locale = False
19 #gathering = implicit
   #gather_subset = all
21 #roles_path    = /etc/ansible/roles
```

4.3 練習系統 ansible.cfg

1. 控制節點 kvm19 有一用户 deyu1，以此用户新增 /home/deyu1/ansible/ansible.cfg，要求如下：

   (a) 指定 inventory 檔案爲 /home/deyu1/ansible/inventory

   (b) 使用於 playbooks 的角色，其位置包含 /home/deyu1/ansible/roles

2. 在 ansible 目錄編輯檔案 ansible.cfg

```
1  [deyu1@kvm19 ansible]$ vim ansible.cfg
```

3. ansible.cfg 檔案依要求設定如下，host_key_checking = False 不使用金鑰檢查，remote_user = root 登入的用户爲 root：

```
[deyu1@kvm19 ansible]$ cat ansible.cfg
[defaults]
inventory      = /home/deyu1/ansible/inventory
roles_path     = /home/deyu1/ansible/roles
host_key_checking = False
remote_user = root
```

4. 新增 /home/deyu1/ansible/roles 目錄。

```
[deyu1@kvm19 ansible]$ mkdir roles
```

4.4 測試 ansible.cfg

1. 查看目錄 /home/deyu1/ansible，目錄下有 ansible.cfg 設定檔及 inventory 檔。

```
[deyu1@kvm19 ansible]$ ll
total 12
-rw-r--r--. 1 deyu1 deyu1   99 Mar  5 09:16 ansible.cfg
-rw-rw-r--. 1 deyu1 deyu1  143 Mar  4 16:29 inventory
drwxrwxr-x. 2 deyu1 deyu1 4096 Mar  5 09:44 roles
```

2. 執行 ansible 列出 webservers 群組的主機，此時會使用工作目錄下的 ansible.cfg 設定檔及 inventory 檔。

```
[deyu1@kvm19 ansible]$ ansible webservers --list-host
  hosts (2):
    kvm13
    kvm14
```

4.5 實機練習

1. 控制節點 kvm19 有一用户 deyu1，以此用户新增 /home/deyu1/ansible/ansible.cfg，要求如下：

   (a) 指定 inventory 檔案爲 /home/deyu1/ansible/inventory

(b) 使用於 playbooks 的角色，其位置包含 /home/deyu1/ansible/roles

(c) 控制節點 ssh 以 root 登入受管節點。

(d) 在 /home/deyu1/inventory 設定所有主機登入密碼變數爲 123qwe。

(e) 控制節點 ssh 登入受管節點，不檢查金鑰。

2. ansible 命令不下 -i 指定 inventory 檔案，列出所有群組主機。

Chapter 5

Ad-hoc 命令

5.1 前言

1. Linux 的文字界面可以輸入簡單命令來完成工作，但對於比較複雜或經常重複的工作，一般會寫成 shell scripts 來完成。
2. Ansible 提供兩種方式在受管節點自動完成任務，一是臨時命令 (ad-hoc command)，另一是寫 ansible playbook。
3. Ansible 的 ad-hoc 命令如同 linux 文字界面命令，playbooks 則類似 shell scripts。
4. ad-hoc 命令適合很少重複執行的任務，重複複雜的任務則由 playbook 來完成。

5.2 *Shell 命令

1. 切換工作目錄到 /home/deyu1/ansible。

```
[deyu1@kvm19 ~]$ cd ansible/
```

2. 執行 ansible ad-hoc 命令，將群組 dev 的主機重開機，出現錯誤。

```
[deyu1@kvm19 ansible]$ ansible dev -a "/sbin/reboot"
kvm11 | FAILED | rc=1 >>
Failed to set wall message, ignoring: Interactive authentication
    required.
Failed to reboot system via logind: Interactive authentication required.
Failed to open initctl fifo: Permission denied
Failed to talk to init daemon.non-zero return code
```

3. 執行 ansible ad-hoc 命令，sudo 將群組 dev 的主機重開機，成功執行，出現的錯誤訊息是因關機時 ssh 無法連線。

```
[deyu1@kvm19 ansible]$ ansible dev -a "sudo /sbin/reboot"
kvm11 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: Shared connection to
kvm11 closed.",
    "unreachable": true
}
```

4. 群組 prod 有 kvm13 及 kvm14 兩台主機都成功重開機，出現的錯誤訊息是因關機時 ssh 無法連線。

```
[deyu1@kvm19 ansible]$ ansible prod -a "sudo /sbin/reboot"
kvm13 | FAILED | rc=-1 >>
Failed to connect to the host via ssh: ssh: connect to host kvm13
 port 22: Connection refused
kvm14 | FAILED | rc=-1 >>
Failed to connect to the host via ssh: ssh: connect to host kvm14
 port 22: Connection refused
```

5. ansible 執行時會同時開啓多個 bash 子程式同步執行，預設數量是 5。

```
[deyu1@kvm19 ansible]$ grep forks /etc/ansible/ansible.cfg
#forks          = 5
```

6. 如果群組的主機很多，可使用選項 -f 指定 bash 子程式數量同時執行，以下是同時開啓 10 個 bash，將受管主機重開機。

```
[deyu1@kvm19 ansible]$ ansible prod -a "sudo /sbin/reboot" -f 10
```

7. -u root 指定以用户 root 執行命令，因爲 deyu1 無法免密碼 ssh 連線 kvm11 的 root 用户，所以出現無法連線的訊息。

```
[deyu1@kvm19 ansible]$ ansible dev -a "/sbin/reboot" -u root
kvm11 | UNREACHABLE! => {
```

```
    "changed": false,
    "msg": "Failed to connect to the host via ssh: root@kvm11:
 Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).",
    "unreachable": true
}
```

8. --ask-pass 要求輸入連線密碼，輸入後成功重新啓動 kvm11，出現的訊息一樣是因關機 ssh 連線中斷。

```
[deyu1@kvm19 ansible]$ ansible dev -a "/sbin/reboot" -u root --ask-pass
SSH password:
kvm11 | FAILED | rc=-1 >>
Failed to connect to the host via ssh: ssh: connect to host kvm11
 port 22: Connection reset by peer
```

5.3 模組文件

1. Ansible-doc 可查詢各模組的文件。

```
[deyu1@kvm19 ansible]$ ansible-doc -h
Usage: ansible-doc [-l|-F|-s] [options] [-t <plugin type> ] [plugin]

plugin documentation tool

Options:
  -h, --help            show this help message and exit
  -j, --json            **For internal testing only** Dump json metadata
      for
                        all plugins.
  -l, --list            List available plugins
  -F, --list_files      Show plugin names and their source files without
                        summaries (implies --list)
  -M MODULE_PATH, --module-path=MODULE_PATH
                        prepend colon-separated path(s) to module
                            library (def
                        ault=~/.ansible/plugins/modules:/usr/share/
                            ansible/plu
                        gins/modules)
  -s, --snippet         Show playbook snippet for specified plugin(s)
  -t TYPE, --type=TYPE  Choose which plugin type (defaults to "module").
                        Available plugin types are : ('become', 'cache',
                        'callback', 'cliconf', 'connection', 'httpapi',
                        'inventory', 'lookup', 'shell', 'module', '
                            strategy',
                        'vars')
  -v, --verbose         verbose mode (-vvv for more, -vvvv to enable
```

```
                        connection debugging)
  --version             show program's version number, config file
      location,
                        configured module search path, module location,
                        executable location and exit

See man pages for Ansible CLI options or website for tutorials
https://docs.ansible.com
```

2. ansible-doc 列出可用的模組

```
[deyu1@kvm19 ansible]$ ansible-doc -l
```

3. ansible-doc 列出可用的模組，過濾 yum，找到有 yum 及 yum_repository 兩個模組。

```
[deyu1@kvm19 ansible]$ ansible-doc -l | grep yum
yum                                                Manages packages
    wit...
yum_repository                                     Add or remove YUM
     re...
```

5.4 檔案傳輸 - copy 模組

1. Ansible 可以用並行的方式同時 scp 大量的檔案到多台主機，完整文件可用 ansible-doc 查詢 copy 模組文件。

```
[deyu1@kvm19  ansible]$ ansible-doc copy
```

2. 先編輯一個測試檔 adhoc.txt

```
[deyu1@kvm19 ansible]$ vim adhoc.txt
[deyu1@kvm19 ansible]$ cat adhoc.txt
ad-hoc file transfer
```

3. 使用 copy 模組將 adhoc.txt 複製到 prod 群組裡所有主機的 /home/deyu1 目錄:

```
[deyu1@kvm19 ansible]$ ansible prod -m copy -a "src=adhoc.txt dest=/home
    /deyu1/"

kvm14 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "checksum": "5ced2367ff60e242577e9a5d90a61233d9336c97",
    "dest": "/home/deyu1/adhoc.txt",
    "gid": 0,
    "group": "root",
    "md5sum": "bb660c5cbe1c8defa638bd8e7592b48a",
    "mode": "0644",
    "owner": "root",
    "secontext": "unconfined_u:object_r:user_home_t:s0",
    "size": 21,
    "src": "/root/.ansible/tmp/ansible-tmp
       -1585003071.5122902-76118385004573/source",
    "state": "file",
    "uid": 0
}
kvm13 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "checksum": "5ced2367ff60e242577e9a5d90a61233d9336c97",
    "dest": "/home/deyu1/adhoc.txt",
    "gid": 0,
    "group": "root",
    "md5sum": "bb660c5cbe1c8defa638bd8e7592b48a",
    "mode": "0644",
    "owner": "root",
    "secontext": "unconfined_u:object_r:user_home_t:s0",
    "size": 21,
    "src": "/root/.ansible/tmp/ansible-tmp
       -1585003071.3513613-64722127143810/source",
    "state": "file",
    "uid": 0
}
```

4. 使用 shell 模組查看 prod 群組裡所有主機，有複製過來的檔案 adhoc.txt，由於 ansible.cfg 設定遠端用户爲 root，所以擁有者及群組皆爲 root。

```
[deyu1@kvm19 ansible]$ ansible prod -m shell -a "ls -l /home/deyu1/adhoc
    .txt"
kvm14 | CHANGED | rc=0 >>
-rw-r--r--. 1 root root 21 Mar 24 06:37 /home/deyu1/adhoc.txt

kvm13 | CHANGED | rc=0 >>
```

```
6 -rw-r--r--. 1 root root 21 Mar 24 06:37 /home/deyu1/adhoc.txt
```

5.5 檔案管理 - file 模組

1. file 模組可以做檔案管理，完整文件可用 ansible-doc 查詢 file 模組文件。

```
[deyu1@kvm19  ansible]$ ansible-doc file
```

2. file 模組檔案管理範例：

   (a) 修改檔案的屬性

```
1 -m file -a "dest=~/foo mode=600"
```

   (b) 修改檔案的擁有者、所屬群組

```
1 -m file -a "dest=~/foo owner=user1 group=shgrp"
```

   (c) 建立目錄，並設定目錄屬性、擁有者、所屬群組。

```
1 -m file -a "dest=~/c mode=755 owner=user1 group=shgrp state=
    directory"
```

   (d) 刪除目錄

```
1 -m file -a "dest=~/c state=absent"
```

3. 使用 file 模組修改檔案 /home/deyu1/adhoc.txt 的屬性成 660，擁有者及群組皆爲 deyu1。

```
1 [deyu1@kvm19 ansible]$ ansible prod -m file -a "dest=/home/deyu1/adhoc.
    txt \
mode=660 owner=deyu1 group=deyu1"
3 kvm13 | CHANGED => {
      "ansible_facts": {
5         "discovered_interpreter_python": "/usr/libexec/platform-python"
```

```
    },
    "changed": true,
    "gid": 1000,
    "group": "deyu1",
    "mode": "0660",
    "owner": "deyu1",
    "path": "/home/deyu1/adhoc.txt",
    "secontext": "unconfined_u:object_r:user_home_t:s0",
    "size": 21,
    "state": "file",
    "uid": 1000
}
kvm14 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "gid": 1000,
    "group": "deyu1",
    "mode": "0660",
    "owner": "deyu1",
    "path": "/home/deyu1/adhoc.txt",
    "secontext": "unconfined_u:object_r:user_home_t:s0",
    "size": 21,
    "state": "file",
    "uid": 1000
}
```

4. 使用 shell 模組查看 prod 群組裡所有主機，adhoc.txt 的屬性已改成 660，擁有者及群組也都改成 deyu1 了。

```
[deyu1@kvm19 ansible]$ ansible prod -m shell -a "ls -l /home/deyu1/adhoc
    .txt"
kvm13 | CHANGED | rc=0 >>
-rw-rw----. 1 deyu1 deyu1 21 Mar 24 06:37 /home/deyu1/adhoc.txt

kvm14 | CHANGED | rc=0 >>
-rw-rw----. 1 deyu1 deyu1 21 Mar 24 06:37 /home/deyu1/adhoc.txt
```

5.6 套件庫 - yum_repository 模組

1. yum_repository 模組可以增加或刪除套件庫，完整文件可用 ansible-doc 查詢 yum_repository 模組文件。

```
[root@kvm19 ~]# ansible-doc yum_repository
```

2. 使用 ad-hoc 命令在每一個受管節點產生 yum repository 設定，並將執行命令寫成腳本 /home/deyu1/ansible/adhoc.sh，yum repository 要求如下：

3. Repository 1

   (a) 名稱 (name)：DYW_BASE

   (b) 描述 (description)：DYW base software

   (c) 套件來源網址：http://dywang.csie.cyut.edu.tw/centos8/BaseOS

   (d) 啓用 GPG 簽章檢查

   (e) GPG 金鑰網址：http://dywang.csie.cyut.edu.tw/materials/RPM-GPG-KEY-centosofficial

4. Repository 2

   (a) 名稱 (name)：DYW_STREAM

   (b) 描述 (description)：DYW stream software

   (c) 套件來源網址：http://dywang.csie.cyut.edu.tw/centos8/AppStream

   (d) 啓用 GPG 簽章檢查

   (e) GPG 金鑰網址：http://dywang.csie.cyut.edu.tw/materials/RPM-GPG-KEY-centosofficial

5. 編輯 /home/deyu1/ansible/adhoc.sh。

```
[deyu1@kvm19 ansible]$ vim adhoc.sh
[deyu1@kvm19 ansible]$ cat adhoc.sh
#!/bin/bash
ansible dev -m yum_repository -a '\
name="DYW_BASE" \
description="DYW base software" \
baseurl="http://dywang.csie.cyut.edu.tw/centos8/BaseOS" \
gpgcheck=yes \
gpgkey="http://dywang.csie.cyut.edu.tw/materials/RPM-GPG-KEY-
    centosofficial" \
enabled=yes'

ansible dev -m yum_repository -a '\
name="DYW_STREAM" \
description="DYW stream software" \
baseurl="http://dywang.csie.cyut.edu.tw/centos8/AppStream" \
gpgcheck=yes \
gpgkey="http://dywang.csie.cyut.edu.tw/materials/RPM-GPG-KEY-
    centosofficial" \
enabled=yes'
```

6. 設定 /home/deyu1/ansible/adhoc.sh 爲可執行。

```
[deyu1@kvm19 ansible]$ chmod +x adhoc.sh
```

7. sudo 執行 /home/deyu1/ansible/adhoc.sh (以下只顯示一台受管主機)。

```
[deyu1@kvm19 ansible]$ ./adhoc.sh
kvm11 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "repo": "DYW_BASE",
    "state": "present"
}
kvm11 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "repo": "DYW_STREAM",
    "state": "present"
}
```

5.7 套件管理 - yum 模組

1. yum 模組可以做套件管理，完整文件可用 ansible-doc 查詢 yum 模組文件。

```
[root@kvm19 ansible]# ansible-doc yum
```

2. 確認 dev 群組裡所有主機，套件 vim 還沒安裝，回傳成功。

```
[deyu1@kvm19 ansible]$ ansible dev -m yum -a "name=vim state=absent"
kvm11 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "msg": "Nothing to do",
    "rc": 0,
    "results": []
}
```

3. 使用 yum 模組安裝 vim 至最新版本，回傳成功，含相依套件共安裝了 4 的套件。

```
[deyu1@kvm19 ansible]$ ansible dev -m yum -a "name=vim state=latest"
kvm11 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
        "Installed: gpm-libs-1.20.7-15.el8.x86_64",
        "Installed: vim-enhanced-2:8.0.1763-10.el8.x86_64",
        "Installed: vim-filesystem-2:8.0.1763-10.el8.noarch",
        "Installed: vim-common-2:8.0.1763-10.el8.x86_64"
    ]
}
```

5.8 實機練習

1. 使用 copy, shell, file 模組，完成以下要求：

   (a) 複製控制主機的 /home/deyu1/adhoc.txt 到節點主機 prod 的 /home/deyu1 目錄。

   (b) 查看複製到節點主機 prod 的 adhoc.txt，其擁有者、所屬群組及檔案權限，爲什麼？

   (c) 將複製到節點主機 prod 的 adhoc.txt 檔案權限改成 660，擁有者及群組都改成 deyu1。

2. 在控制主機以 deyu1 用户寫一腳本 /home/deyu1/ansible/adhoc.sh，在每一個受管節點產生 yum repository 設定，要求如下：

   (a) Repository 1

      i. 名稱 (name)：DYW_BASE

      ii. 描述 (description)：DYW base software

      iii. 套件來源網址：http://dywang.csie.cyut.edu.tw/centos8/BaseOS

      iv. 啓用 GPG 簽章檢查

      v. GPG 金鑰網址：http:// dywang.csie.cyut.edu.tw/ materials/ RPM-GPG-KEY-centosofficial

(b) Repository 2

i. 名稱 (name)：DYW_STREAM

ii. 描述 (description)：DYW stream software

iii. 套件來源網址：http://dywang.csie.cyut.edu.tw/centos8/AppStream

iv. 啓用 GPG 簽章檢查

v. GPG 金鑰網址：http:// dywang.csie.cyut.edu.tw/ materials/ RPM-GPG-KEY-centosofficial

3. 使用 yum 模組，在主機 test 安裝最新版的 tar 套件。

Chapter 6

Playbooks

6.1 前言

1. 相較於 ad-hoc 命令，Playbooks 是比較完整且功能強的 ansible 配置方法。
2. Ansible 的腳本 (playbook) 是用 YAML (YAML Ain't Markup Language) 語言撰寫透過編寫腳本告訴 Ansible 需要做的事項。
3. YAML (Yet Another Markup Language)，原意是另一種標記語言。
4. 爲了強調 YAML 語言以資料爲中心，而不是以標記語言爲重點，而用反向縮寫重新命名。YAML (YAML Ain't a Markup Language)，不是一種標記語言。
5. YAML 以空白符號 (不能用 TAB) 縮排來編輯資料，適合表達階層式資料結構。
6. YAML 符號

```
1 ---       分隔檔案
  -         減號+空白：清單 (陣列)
```

7. YAML 具有高度可讀性特性，不會像是程式語言或指令碼，所以一份好的 Ansible playbook 是容易理解及修改維護。

6.2 第一支 playbook

1. 切換工作目錄到 /home/deyu1/ansible。

```
[deyu1@kvm19 ~]$ cd ansible/
```

2. 在工作目錄建立 playbook.yml，對主機 dev 定義了 2 個任務：

   (a) 名爲"test connection" 的任務，呼叫 ansible 內建模組 ping 測試連線，再透過 register 模組把回傳訊息存到變數 message。

   (b) 名爲"print debug message" 的任務，呼叫 debug 模組把變數 message 內容輸出到終端機。

   (c) 在 Ansible 中要取變數內容，必須在變數外加兩個大括號 {{}}。

```
[deyu1@kvm19 ansible]$ vim playbook.yml
[deyu1@kvm19 ansible]$ cat playbook.yml
---
- hosts: dev
  tasks:
    # task 1
    - name: test connection
      ping:
      register: message

    # task 2
    - name: print debug message
      debug:
        msg: "{{ message }}"
```

3. 使用 ansible-playbook 執行 playbook.xml 腳本，成功連線 kvm11，且印出訊息。

```
[deyu1@kvm19 ansible]$ ansible-playbook playbook.yml

PLAY [dev]
    ********************************************************************


TASK [Gathering Facts]
    ***********************************************************
ok: [kvm11]

TASK [test connection]
    ***********************************************************
ok: [kvm11]

TASK [print debug message]
    ***************************************************
ok: [kvm11] => {
    "msg": {
        "changed": false,
        "failed": false,
        "ping": "pong"
    }
```

```
18 }

20 PLAY RECAP
    *********************************************************************

kvm11 : ok=3 changed=0 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

6.3 基本語法

1. 看到 playbook 腳本中關鍵字 (Keywords) 不清楚時，可上 ansible 官網查詢。

2. 最基本的 playbook 分成三個部分：

   (a) 在哪些主機以什麼身份執行，例如：

```
1 hosts: dev
  user: root
```

   (b) 執行的任務，notify 當任務執行通知"restart httpd"，例如：

```
  tasks:
2   - name: install httpd
      yum: pkg=httpd state=latest
4     notify: restart httpd
```

   (c) 善後任務，例如：

```
  handlers:
2   - name: restart httpd
      service: name=httpd state=restarted
```

3. 以上基本語法寫成腳本 deploy.yml，其中 notify 當任務執行通知 headlers " restart httpd"。

```
1 [deyu1@kvm19 ansible]$ vim deploy.yml
  [deyu1@kvm19 ansible]$ cat deploy.yml
3 ---
  - hosts: dev
5   tasks:
```

```
  - name: install httpd
    yum: pkg=httpd state=latest
    notify: restart httpd

  handlers:
  - name: restart httpd
    service: name=httpd enabled=yes state=restarted
```

4. 使用 ansible-playbook 執行 httpddeploy.xml 腳本，成功安裝並執行善後工作，設定開機啓動 httpd 及現在重啓。

```
[deyu1@kvm19 ansible]$ ansible-playbook deploy.yml

PLAY [dev]
    ********************************************************************


TASK [Gathering Facts]
    ***********************************************************
ok: [kvm11]

TASK [install httpd]
    ************************************************************
changed: [kvm11]

RUNNING HANDLER [restart httpd]
    ***********************************************
changed: [kvm11]

PLAY RECAP
    ********************************************************************

kvm11 : ok=3 changed=2 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

5. 使用 ad-hoc 命令檢查 dev 的 httpd 服務是否啓動？回傳"changed": false, 表示服務已經啓動，沒有做任何改變。

```
[deyu1@kvm19 ansible]$ ansible dev -m service -a "name=httpd state=
    started" --check
kvm11.deyu.wang | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "name": "httpd",
    "state": "started",
    "status": {
```

```
.....
```

6. 使用 ad-hoc 命令檢查 dev 的 httpd 服務是否開機啓動 (enabled)？回傳"changed": false, 表示服務已經設定開機啓動，沒有做任何改變。

```
[deyu1@kvm19 ansible]$ ansible dev -m service -a "name=httpd enabled=yes
    " --check
kvm11.deyu.wang | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "enabled": true,
    "name": "httpd",
    "status": {
.....
```

6.4 使用變數

1. 腳本 deploy1.yml，在 dev 群組主機安裝套件，套件名稱爲變數 item。

```
[deyu1@kvm19 ansible]$ vim deploy1.yml
[deyu1@kvm19 ansible]$ cat deploy1.yml
---
- hosts: dev
  tasks:
  - name: install package
    yum: pkg={{item}} state=latest
```

2. 命令列使用 ansible-playbook 執行 deploy1.xml 腳本時，使用 -extra-vars 提供變數"item=php"，成功安裝 php 套件。

```
[deyu1@kvm19 ansible]$ ansible-playbook deploy1.yml --extra-vars "item=
    php"

PLAY [dev]
    ********************************************************************



TASK [Gathering Facts]
    *********************************************************
ok: [kvm11]

```

```
TASK [install package]
    ***********************************************************
changed: [kvm11]

PLAY RECAP
    *********************************************************************

kvm11 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

3. 使用 ad-hoc 命令安裝 dev 的 php 套件，回傳"changed": false, 表示 php 套件已經安裝，沒有做任何改變。

```
[deyu1@kvm19 ansible]$ ansible dev -m yum -a "name=php state=present"
kvm11.deyu.wang | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "msg": "Nothing to do",
    "rc": 0,
    "results": []
}
```

6.5 loop 迴圈

1. ansible 2.5 前使用 with_* 關鍵字，例如 with_items 來運作迴圈。ansible 2.6 版後官方推薦 loop 關鍵字代替 with_*，一般而言，關鍵字 with_* 大多可用 loop 代替，兩者差異及迴圈詳細用法請參考 ansible 官方文件。

2. loop 類似程式的 for 迴圈，腳本 deploy2.yml，在 dev 群組主機安裝套件，套件名稱變數 item，此爲 loop 迴圈的預設變數，loop 迴圈列出 php 及 mariadb 兩個套件。

```
[deyu1@kvm19 ansible]$ vim deploy2.yml
[deyu1@kvm19 ansible]$ cat deploy2.yml
---
- hosts: dev
  tasks:
  - name: install package
    yum: pkg={{item}} state=latest
    loop:
    - php
    - mariadb
```

3. 命令列使用 ansible-playbook 執行 deploy2.xml 腳本，回傳 php 已經安裝，mariadb 成功安裝。

```
[deyu1@kvm19 ansible]$ ansible-playbook deploy2.yml

PLAY [dev]
    ***********************************************************************


TASK [Gathering Facts]
    ************************************************************
ok: [kvm11]

TASK [install package]
    ************************************************************
ok: [kvm11] => (item=php)
changed: [kvm11] => (item=mariadb)

PLAY RECAP
    ***********************************************************************

kvm11 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

4. deploy2.yml 使用 loop 提供 yum 安裝套件名稱，每個套件安裝都獨立執行 yum。目前的 ansible 版本建議改用 yum 模組內建的 name，一次指定多個套件，如下 deploy2a.yml。

```
[deyu1@kvm19 ansible]$ cat deploy2a.yml
---
- hosts: dev
  user: root
  tasks:
  - name: install package
    yum:
      name:
      - php
      - mariadb
      state: latest
```

5. 命令列使用 ansible-playbook 執行 deploy2a.xml 腳本，回傳 php 已經安裝，mariadb 成功安裝。

```
[deyu1@kvm19 ansible]$ ansible-playbook deploy2a.yml

PLAY [dev]
    ********************************************************************


TASK [Gathering Facts]
    ************************************************************
ok: [kvm11]

TASK [install package]
    ************************************************************
ok: [kvm11]

PLAY RECAP
    ********************************************************************

kvm11 : ok=2 changed=0 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

6.6 block 區塊

1. 當定義的任務只有符合條件的群組或主機才需要執行時，會把這部分任務定義成 block，配合 when 條件判斷，決定是否執行此 block。

2. 腳本 deploy3.yml，在 dev 及 test 群組主機安裝套件，block 定義安裝 Development Tools 群組套件，when 內縮位置對齊 block，表示條件 test 在 group_names 時才執行 block，group_names 變數是指目前執行的主機群組。

```
[deyu1@kvm19 ansible]$ vim deploy3.yml
[deyu1@kvm19 ansible]$ cat deploy3.yml
---
- hosts: dev test
  tasks:
  - name: install package
    yum:
      name:
      - php
      - mariadb
      state: latest
  - block:
    - name: install the '@RPM Development Tools' package group
      yum:
        name: "@RPM Development Tools"
        state: present
    when: "'test' in group_names"
```

3. 命令列使用 ansible-playbook 執行 deploy3.xml 腳本，回傳 kvm11 已安裝 php 及 mariadb，kvm12 成功安裝 php 及 mariadb。block 部分，kvm11 跳過，kvm12 成功安裝 Development Tools 群組套件。

```
[deyu1@kvm19 ansible]$ ansible-playbook deploy3.yml

PLAY [dev test]
    ********************************************************************

TASK [Gathering Facts]
    ***********************************************************
ok: [kvm11]
ok: [kvm12]

TASK [install package]
    ***********************************************************
ok: [kvm11]
changed: [kvm12]

TASK [install the '@RPM Development Tools' package group]
    **********************
skipping: [kvm11.deyu.wang]
changed: [kvm12]

PLAY RECAP
    ********************************************************************

kvm11 : ok=2 changed=0 unreachable=0 failed=0 skipped=1 rescued=0
    ignored=0
kvm12 : ok=3 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

6.7 更新套件

1. 修改腳本 deploy3.yml 爲 deploy4.yml，增加任務在 test 更新 open 開頭的套件。

```
[deyu1@kvm19 ansible]$ cat deploy4.yml
---
- hosts: dev test
  tasks:
  - name: install package
    yum:
      name:
      - php
      - mariadb
      state: latest
  - block:
```

```
    - name: install the '@RPM Development Tools' package group
      yum:
        name: "@RPM Development Tools"
        state: present
    - name: upgrade all packages
      yum:
        name: 'open*'
        state: latest
    when: "'test' in group_names"
```

2. 命令列使用 ansible-playbook 執行 deploy4.xml 腳本。

```
[deyu1@kvm19 ansible]$ ansible-playbook deploy4.yml

PLAY [dev test]
    ********************************************************************

TASK [Gathering Facts]
    ************************************************************
ok: [kvm12]
ok: [kvm11]

TASK [install package]
    ************************************************************
ok: [kvm11]
ok: [kvm12]

TASK [install the '@RPM Development Tools' package group]
    **********************
skipping: [kvm11]
ok: [kvm12]

TASK [upgrade all packages]
    ***************************************************
skipping: [kvm11]
ok: [kvm12]

PLAY RECAP
    ********************************************************************************

kvm11 : ok=2 changed=0 unreachable=0 failed=0 skipped=2 rescued=0
    ignored=0
kvm12 : ok=4 changed=0 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

6.8 實機練習

1. 共同要求：執行 ansible-playbook 時，終端機寬度限定 80。

2. 寫 yaml 腳本 pbook1.yml，對 dev 主機要求如下：
   (a) 使用 shell 模組執行”ls -l /bin/passwd”。
   (b) ansible-playbook 執行結果導向到 /home/deyu1/ansible/pbook1.txt。
3. 寫 yaml 腳本 pbook2.yml，對 dev 主機要求如下：
   (a) yum 模組移除 xz 套件
   (b) 移除後 notify ”install xz”
   (c) handlers 使用 yum 模組再安裝 xz
   (d) ansible-playbook 執行結果導向到 /home/deyu1/ansible/pbook2.txt。
4. 修改 pbook2.yml 爲 pbook3.yml，對 dev 主機要求如下：
   (a) yum 模組移除的套件改成變數 item
   (b) 移除後 notify ”install item”
   (c) handlers 使用 yum 模組再安裝 item
   (d) ansible-playbook 執行時使用 `--extra-vars` 指定套件名稱。
5. 寫 yaml 腳本 pbook4.yml，對 dev 主機要求如下：
   (a) 使用 loop 迴圈。
   (b) shell 模組 ls 列出 loop 迴圈的檔案，檔名由外加變數 f1 及 f2 決定。
6. 寫 yaml 腳本 pbook5.yml，對 dev 及 test 群組主機要求如下：
   (a) 安裝 xz 及 gzip 兩個套件
   (b) 使用 block ... when 語法，dev 才安裝 vim 套件。
7. 修改 pbook5.yml 爲 pbook6.yml，增加只有 dev 主機才更新 tar 套件的任務。

Chapter 7

Jinja2 樣板

7.1 前言

1. 如果在 100 台主機安裝 httpd 服務，每台的歡迎網頁要顯示主機 IP 或名稱，如何解決？
2. ansible 使用 jinja2 樣板修改受管主機的配置文件，可解決此問題。
3. 使用 jinja2 樣板也就是使用 template 模組，template 模組和 copy 模組一樣，都是將檔案複製到遠端主機，不同的是，template 模組可以取得檔案中的變數，而 copy 則是原封不動的把檔案複製過去。
4. Ansible 還允許 jinja2 樣板使用條件判斷和迴圈，但不允許在 playbook 中使用。

7.2 基本語法

1. 輸出自定義或 facts 變數值。

```
{{ EXPR }}
```

2. 註解

```
{# COMMENT #}
```

3. 邏輯判斷

```
{% if EXPR %}
{% elif EXPR %}
{% else %}
{% endif %}
```

4. for 迴圈

```
{% for i in EXPR %}
{% endfor %}
```

7.3 變數

1. 合法變數名：可以是字母、數字或下底線，例如：

```
foo_port
foo7
```

2. Magic 變數：ansible 保留變數，user 不能設定，完整變數說明請查看 ansible 文件。

```
ansible_check_mode
ansible_dependent_role_names
ansible_diff_mode
ansible_forks
ansible_inventory_sources
ansible_limit
ansible_loop
ansible_loop_var
ansible_play_batch (D)
ansible_play_hosts (D)
ansible_play_hosts_all
ansible_play_role_names
ansible_playbook_python
ansible_role_names
ansible_run_tags
ansible_search_path
ansible_skip_tags
ansible_verbosity
ansible_version
group_names
groups
hostvars
inventory_hostname
```

```
inventory_hostname_short
inventory_dir
inventory_file
omit
play_hosts (D)
ansible_play_name
playbook_dir
role_name
role_names
role_path
```

3. Facts 變數：不是 magic 變數，執行 setup 模組時收集，包含目前主機 (inventory_hostname) 的相關訊息，受管主機系統不同變數內容也會不同，不過變數都以 ansible_ 開頭。

```
ansible_facts
ansible_local
ansible_*
```

7.4 debug 輸出 magic 變數

1. debug 模組列印變數 groups，內容由 inventory 設定取得。

```
[deyu1@kvm19 ansible]$ ansible -m debug dev -a "var=groups"
kvm11 | SUCCESS => {
    "groups": {
        "all": [
            "kvm11",
            "kvm12",
            "kvm15",
            "kvm13",
            "kvm14"
        ],
        "balancers": [
            "kvm15"
        ],
        "dev": [
            "kvm11"
        ],
        "prod": [
            "kvm13",
            "kvm14"
        ],
        "test": [
            "kvm12"
        ],
```

```
        "ungrouped": [],
        "webservers": [
            "kvm13",
            "kvm14"
        ]
    }
}
```

2. debug 模組列印變數 group_names，內容由 inventory 設定取得。

```
[deyu1@kvm19 ansible]$ ansible -m debug all -a "var=group_names"
kvm11 | SUCCESS => {
    "group_names": [
        "dev"
    ]
}
kvm12 | SUCCESS => {
    "group_names": [
        "test"
    ]
}
kvm15 | SUCCESS => {
    "group_names": [
        "balancers"
    ]
}
kvm13 | SUCCESS => {
    "group_names": [
        "prod",
        "webservers"
    ]
}
kvm14 | SUCCESS => {
    "group_names": [
        "prod",
        "webservers"
    ]
}
```

3. debug 模組列印變數 hostvars['kvm11']，ansbile_facts 是空的，因爲 debug 模組沒有執行 setup 模組。

```
[deyu1@kvm19 ansible]$ ansible -m debug dev -a "var=hostvars['kvm11']"
kvm11 | SUCCESS => {
    "hostvars['kvm11']": {
        "ansible_check_mode": false,
        "ansible_diff_mode": false,
        "ansible_facts": {},
        "ansible_forks": 5,
```

```
        "ansible_inventory_sources": [
            "/home/deyu1/ansible/inventory"
        ],
        "ansible_playbook_python": "/usr/libexec/platform-python",
        "ansible_ssh_host": "192.168.122.11",
        "ansible_verbosity": 0,
        "ansible_version": {
            "full": "2.8.5",
            "major": 2,
            "minor": 8,
            "revision": 5,
            "string": "2.8.5"
        },
        "group_names": [
            "dev"
        ],
        "groups": {
            "all": [
                "kvm11",
                "kvm12",
                "kvm15",
                "kvm13",
                "kvm14"
            ],
            "balancers": [
                "kvm15"
            ],
            "dev": [
                "kvm11"
            ],
            "prod": [
                "kvm13",
                "kvm14"
            ],
            "test": [
                "kvm12"
            ],
            "ungrouped": [],
            "webservers": [
                "kvm13",
                "kvm14"
            ]
        },
        "inventory_dir": "/home/deyu1/ansible",
        "inventory_file": "/home/deyu1/ansible/inventory",
        "inventory_hostname": "kvm11",
        "inventory_hostname_short": "kvm11",
        "omit": "
            __omit_place_holder__354579cf702a43d1da4b8c98248aa4acbce888dc
            ",
        "playbook_dir": "/home/deyu1/ansible"
    }
}
```

7.5 setup 輸出 facts 變數

1. setup 模組可以列印所有 ansible_facts 變數內容。

```
[deyu1@kvm19 ansible]$ ansible dev -m setup
```

2. setup 模組列印 ansible_facts 變數，增加參數 filter=ansible_default_ipv4，取得其內容。

```
[deyu1@kvm19 ansible]$ ansible dev -m setup -a "filter=
    ansible_default_ipv4"
kvm11 | SUCCESS => {
    "ansible_facts": {
        "ansible_default_ipv4": {
            "address": "192.168.122.11",
            "alias": "ens3",
            "broadcast": "192.168.122.255",
            "gateway": "192.168.122.1",
            "interface": "ens3",
            "macaddress": "52:54:00:18:6e:94",
            "mtu": 1500,
            "netmask": "255.255.255.0",
            "network": "192.168.122.0",
            "type": "ether"
        },
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false
}
```

3. setup 模組列印 ansible_facts 變數，增加參數 filter=ansible_fqdn，取得其內容。

```
[deyu1@kvm19 ansible]$ ansible dev -m setup -a "filter=ansible_fqdn"
kvm11 | SUCCESS => {
    "ansible_facts": {
        "ansible_fqdn": "kvm11.deyu.wang",
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false
}
```

4. setup 模組列印 ansible_facts 變數，增加參數 filter=ansible_hostname，取得其內容。

```
[deyu1@kvm19 ansible]$ ansible dev -m setup -a "filter=ansible_hostname"
kvm11 | SUCCESS => {
    "ansible_facts": {
        "ansible_hostname": "kvm11",
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false
}
```

7.6 產生 hosts 檔案

1. 下載 hosts.j2 樣板檔。

2. host2.j2 格式爲 IP FQDN HOSTNAME (FQDN, Fully qualified domain name, 完全資格域名)，修改 host2.j2 產生所有 inventroy hosts。

```
[deyu1@kvm19 ansible]$ vim hosts.j2
[deyu1@kvm19 ansible]$ cat hosts.j2
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.
    localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain

{% for host in groups['all'] %}
{{ hostvars[host]['ansible_default_ipv4']['address'] }} {{ hostvars[host
    ]
['ansible_fqdn'] }} {{ hostvars[host]['ansible_hostname'] }}
{% endfor %}
```

3. 建立 playbook 腳本 hosts.yml，當主機在 dev 群組時，使用 hosts.j2 樣板產生 /etc/myhosts。

```
[deyu1@kvm19 ansible]$ vim hosts.yml
[deyu1@kvm19 ansible]$ cat hosts.yml
---
- name: Create /etc/myhosts
  hosts: all
  tasks:
  - name: Template a file to /etc/myhosts
    template:
      src: /home/deyu1/ansible/hosts.j2
      dest: /etc/myhosts
    when: '"dev" in group_names'
```

7.7 測試 hosts.yml

1. ansible-playbook 執行 host2.yml 腳本，成功在 dev 主機 kvm11 產生 /etc/myhosts。

```
[deyu1@kvm19 ansible]$ ansible-playbook hosts.yml

PLAY [Create /etc/myhosts]
    ****************************************************

TASK [Gathering Facts]
    ************************************************************
ok: [kvm15]
ok: [kvm12]
ok: [kvm11]
ok: [kvm13]
ok: [kvm14]

TASK [Template a file to /etc/myhosts]
    ***************************************
skipping: [kvm12]
skipping: [kvm15]
skipping: [kvm13]
skipping: [kvm14]
changed: [kvm11]

PLAY RECAP
    *********************************************************************

kvm11 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
kvm12 : ok=1 changed=0 unreachable=0 failed=0 skipped=1 rescued=0
    ignored=0
kvm13 : ok=1 changed=0 unreachable=0 failed=0 skipped=1 rescued=0
    ignored=0
kvm14 : ok=1 changed=0 unreachable=0 failed=0 skipped=1 rescued=0
    ignored=0
kvm15 : ok=1 changed=0 unreachable=0 failed=0 skipped=1 rescued=0
    ignored=0
```

2. 查詢 kvm11 的 /etc/myhosts，群組所有主機皆已依格式列出。

```
[deyu1@kvm19 ansible]$ ansible dev -m shell -a "cat /etc/myhosts"
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.
    localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain

192.168.122.11 kvm11.deyu.wang kvm11
192.168.122.12 kvm12.deyu.wang kvm12
192.168.122.15 kvm15.deyu.wang kvm15
```

```
8 192.168.122.13 kvm13.deyu.wang kvm13
  192.168.122.14 kvm14.deyu.wang kvm14
```

7.8 實機練習

1. 下載 hosts.j2 樣板檔。
2. 修改 hosts.j2
   (a) 增加所有受管主機名稱與 IP 對應
   (b) 對應格式爲：`IPV4 FQDN inventory_hostname`
   (c) 格式中的三項都使用 ansible 變數取得
3. 撰寫 yaml 腳本 hosts1.yml，要求如下：
   (a) 對所有受管主機執行任務。
   (b) 只有 test 主機時，使用 hosts.j2 樣板產生 /opt/myhosts。
4. 修改 hosts.j2 存成 hosts2.j2
   (a) for 迴圈中增加 if 判斷
   (b) 如果是 prod 主機，則輸出 IP FQDN prod (prod 是字串)。
5. 修改 hosts1.yml 存成 hosts2.yml
   (a) 使用 hosts2.j2
   (b) 目標檔案改成 /opt/myhosts2。

Chapter 8

Roles

8.1 前言

1. 很多時候希望有部分的部署內容是可以被其他不同的 playbook 重新使用，但卻不希望在每一個不同的 playbook 中都要重新定義一次。
2. Ansible 提供了角色 (role) 的概念，可以透過撰寫客製的 role 來讓所有 playbook 重複使用。
3. role 根據檔案結構自動載入變數檔 (`vars_files`)、任務 (tasks)、善後程序 (handlers)。

8.2 項目結構

1. role 必須至少包含以下目錄之一，但最好排除任何未使用的目錄。
   (a) 任務 (tasks)：role 要執行的任務列表。
   (b) 善後程序 (handlers)：該角色甚至該角色之外的任何地方都可以使用這些善後程序。
   (c) 預設 (defaults)：role 預設的變數。
   (d) vars：role 的其他變數。
   (e) 檔案 (files)：可以透過此角色部署的檔案。
   (f) 樣板 (templates)：可以透過此角色部署的樣板。
   (g) meta：爲此角色定義一些 meta data。
2. roles 目錄結構範例：roles 目錄下有 common 及 webservers 兩個 roles。

```
roles/
    common/
        tasks/
        handlers/
        files/
        templates/
        vars/
        defaults/
        meta/
    webservers/
        tasks/
        defaults/
        meta/
```

3. 使用時，每個目錄必須包含一個 main.yml 檔案。

8.3 初始化角色 apache

1. 建立 roles 目錄。

```
[deyu1@kvm19 ansible]$ mkdir roles/
```

2. 查看 ansible.cfg 配置檔的 roles_path，是否已設定 /home/deyu1/ansible/roles 目錄。

```
[deyu1@kvm19 ansible]$ vim ansible.cfg
[deyu1@kvm19 ansible]$ grep ^roles_path ansible.cfg
roles_path    = /home/deyu1/ansible/roles
```

3. 進入 roles 目錄。

```
[deyu1@kvm19 ansible]$ cd roles/
[deyu1@kvm19 roles]$ pwd
/home/deyu1/ansible/roles
```

4. ansible-galaxy 初使化 apache 角色。

```
[deyu1@kvm19 roles]$ ansible-galaxy init apache
- apache was created successfully
```

5. 查看 apache 目錄，已建立角色需要的目錄及檔案。

```
[deyu1@kvm19 roles]$ tree apache/
apache/
    defaults
        main.yml
    files
    handlers
        main.yml
    meta
        main.yml
    README.md
    tasks
        main.yml
    templates
    tests
        inventory
        test.yml
    vars
        main.yml

8 directories, 8 files
```

6. 回到 ansible 目錄，查看角色，列出 apache 角色。

```
[deyu1@kvm19 ansible]$ cd ..
[deyu1@kvm19 ansible]$ ansible-galaxy list
# /home/deyu1/ansible/roles
- apache, (unknown version)
```

8.4 編輯角色 apache 任務

1. 編輯任務主檔 main.yml，新增任務：

   (a) 安裝最新版本的 httpd

   (b) 設定 htppd 開機啓動且立即啓動

   (c) 防火牆開放 http 服務

```
[deyu1@kvm19 ansible]$ vim roles/apache/tasks/main.yml
[deyu1@kvm19 ansible]$ cat roles/apache/tasks/main.yml
---
- name: install the latest version of Apache
  yum:
    name: httpd
```

```
    state: latest

- name: Start service httpd
  service:
    name: httpd
    state: started
    enabled: yes

- name: Enable service firewalld
  service:
    name: firewalld
    enabled: yes
    state: started

- name: firewalld add http service
  firewalld:
    service: http
    permanent: yes
    state: enabled
    immediate: yes
```

8.5 編輯角色 apache 樣板

1. 在樣板目錄 templates 新增樣板檔案 index.html.j2，其中變數 ansible_fqdn

```
[deyu1@kvm19 ansible]$ vim roles/apache/templates/index.html.j2
[deyu1@kvm19 ansible]$ cat roles/apache/templates/index.html.j2
Welcome to {{ ansible_fqdn }} on {{ ansible_default_ipv4.address }}
```

2. 其中主機名稱及 IP 可用 ansible -m setup 查詢，過濾字串 ansible_fqdn。

```
[deyu1@kvm19 ansible]$ ansible kvm14 -m setup | grep fqdn
        "ansible_fqdn": "kvm14.deyu.wang",
[deyu1@kvm19 ansible]$ ansible kvm14 -m setup -a "filter=ansible_fqdn"
kvm14 | SUCCESS => {
    "ansible_facts": {
        "ansible_fqdn": "kvm14.deyu.wang",
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false
}
```

3. 其中主機名稱及 IP 可用 ansible -m setup 查詢，過濾字串 default_ipv4。

```
[deyu1@kvm19 ansible]$ ansible kvm14 -m setup | grep ipv4
        "ansible_all_ipv4_addresses": [
        "ansible_default_ipv4": {
                "tx_checksum_ipv4": "on",
            "ipv4": {
                "tx_checksum_ipv4": "off [fixed]",
            "ipv4": {
[deyu1@kvm19 ansible]$ ansible kvm14 -m setup -a "filter=
    ansible_default_ipv4"
kvm14 | SUCCESS => {
    "ansible_facts": {
        "ansible_default_ipv4": {
            "address": "192.168.122.14",
            "alias": "ens3",
            "broadcast": "192.168.122.255",
            "gateway": "192.168.122.1",
            "interface": "ens3",
            "macaddress": "52:54:00:59:28:59",
            "mtu": 1500,
            "netmask": "255.255.255.0",
            "network": "192.168.122.0",
            "type": "ether"
        },
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false
}
```

4. 在任務主檔 main.yml 新增任務，將樣板 index.html.j2 複製到 /var/www/html/index.html。

```
[deyu1@kvm19 ansible]$ vim roles/apache/tasks/main.yml
[deyu1@kvm19 ansible]$ cat roles/apache/tasks/main.yml
---
- name: install the latest version of Apache
  yum:
    name: httpd
    state: latest
- name: Start service httpd
  service:
    name: httpd
    state: started
    enabled: yes

- name: Enable service firewalld
  service:
    name: firewalld
    enabled: yes
    state: started

- name: firewalld add http service
```

```
  firewalld:
22    service: http
    permanent: yes
24    state: enabled
    immediate: yes
26
- name: Template a file
28  template:
    src: index.html.j2
30    dest: /var/www/html/index.html
```

5. 再查看 roles/apache 目錄，樣板目錄 templates 下有 index.html.j2：

```
[deyu1@kvm19 ansible]$ tree roles/apache/
2 roles/apache/
    defaults
4       main.yml
    files
6   handlers
        main.yml
8   main.yml
    meta
10      main.yml
    README.md
12  tasks
        main.yml
14  templates
        index.html.j2
16  tests
        inventory
18      test.yml
    vars
20      main.yml

22 8 directories, 10 files
```

8.6 使用角色 apache

1. 建立 playbook apache.yml，群組 webservers 使用角色 apache。

```
[deyu1@kvm19 ansible]$ vim apache.yml
2 [deyu1@kvm19 ansible]$ cat apache.yml
---
4 - name: use role
  hosts: webservers
6 roles:
  - apache
```

2. 執行 playbook apache.yml，成功完成任務。

```
[deyu1@kvm19 ansible]$ ansible-playbook apache.yml

PLAY [use role]
    ********************************************************************

TASK [Gathering Facts]
    ************************************************************
ok: [kvm14]
ok: [kvm13]

TASK [apache : install the latest version of Apache]
    ***************************
changed: [kvm14]
changed: [kvm13]

TASK [apache : Start service httpd]
    *****************************************
changed: [kvm14]
changed: [kvm13]

TASK [apache : firewalld]
    *****************************************************
changed: [kvm14]
changed: [kvm13]

TASK [apache : Template a file]
    ************************************************
changed: [kvm14]
changed: [kvm13]

PLAY RECAP
    ********************************************************************************

kvm13 : ok=5 changed=4 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
kvm14 : ok=5 changed=4 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

3. 測試連線。

```
[deyu1@kvm19 ansible]$ curl http://kvm14/
Welcome to kvm14.deyu.wang on 192.168.122.14
[deyu1@kvm19 ansible]$ curl http://kvm13/
Welcome to kvm13.deyu.wang on 192.168.122.13
```

8.7 實機練習

1. 初使化一個 nfs 角色。
2. 編輯 nfs 角色，任務如下：
   (a) 安裝 nfs-utils 套件
   (b) 設定開機啓動 nfs-server 服務，且現在啓動。
   (c) 啓用 firewalld
   (d) firewalld 服務開放 nfs, rpc-bind, mountd 等服務，開啓三個服務要使用 loop 迴圈。

```
  firewalld:
    service: "{{ item }}"
    permanent: yes
    state: enabled
    immediate: yes
  loop:
    - nfs
    - rpc-bind
    - mountd
```

   (e) 建立分享目錄 /public，提示：file 模組範例。

```
- name: Create a directory if it does not exist
  file:
    path: /etc/some_directory
    state: directory
    mode: '0755'
```

   (f) 新增樣板 exports.j2，並複製到 /etc/exports。
      i. 樣板檔案 exports.j2 內容如下，其中分享網段 xxx.xxx.xxx.0 由變數 ansible_default_ipv4.network 取得。

```
/public xxx.xxx.xxx.0/24(rw,sync)
```

      ii. 目標位置：/etc/exports
      iii. 複製樣板檔案後，必須重啓 nfs-server 服務才會生效，所以 nfs-server 服務的狀態要使用"restarted"，且 templates 模組也必須放在 nfs-server 重啓任務之前。

```
1 showmount -e kvm11
```

3. 建立 nfs.yml 腳本，對 dev 主機執行任務，要求如下：

   (a) 使用 nfs 角色

   (b) 完成任務後，showmount 查詢 dev 主機是否分享 /public 目錄？

Chapter 9

RHEL Roles

9.1 前言

1. 除了自訂角色提供不同的 playbook 使用外，RHEL 已建立了幾個常用的 system roles。
2. RHEL system roles 有 kdump, network, postfix, selinux, timesync 等五個角色。

9.2 安裝

1. Ansible 要使用 RHEL system roles，必須先安裝 rhel-system-roles 套件。

```
[deyu1@kvm19 ansible]$ sudo yum install rhel-system-roles.noarch
```

2. /usr/share/ansible/roles 目錄下有五個 rhel-system-roles。

```
[deyu1@kvm19 ansible]$ ls -d /usr/share/ansible/roles/rhel-system-roles
    .*
/usr/share/ansible/roles/rhel-system-roles.kdump
/usr/share/ansible/roles/rhel-system-roles.network
/usr/share/ansible/roles/rhel-system-roles.postfix
/usr/share/ansible/roles/rhel-system-roles.selinux
/usr/share/ansible/roles/rhel-system-roles.timesync
```

3. 查看 rhel-system-role timesync 的目錄結構。

```
[deyu1@kvm19 ~]$ tree /usr/share/ansible/roles/rhel-system-roles.
    timesync
```

```
/usr/share/ansible/roles/rhel-system-roles.timesync
    COPYING
    defaults
        main.yml
    handlers
        main.yml
    library
        timesync_provider.sh
    meta
        main.yml
    README.html
    README.md
    semaphore
    tasks
        main.yml
    templates
        chrony.conf.j2
        chronyd.sysconfig.j2
        ntp.conf.j2
        ntpd.sysconfig.j2
        phc2sys.sysconfig.j2
        ptp4l.conf.j2
        ptp4l.sysconfig.j2
        timemaster.conf.j2
    tests
        roles
            timesync -> ../..
        tests_default.yml
        tests_ntp_provider1.yml
        tests_ntp_provider2.yml
        tests_ntp_provider3.yml
        tests_ntp_provider4.yml
        tests_ntp_provider5.yml
        tests_ntp_ptp.yml
        tests_ntp.yml
        tests_ptp_multi.yml
        tests_ptp_single.yml
    vars
        main.yml

10 directories, 28 files
```

9.3 RHEL 角色目錄

1. 要使用 RHEL 系統角色，必須先將其 roles 目錄位置加到目前工作目錄下的 ansible.cfg。

```
[deyu1@kvm19 ansible]$ vim ansible.cfg
[deyu1@kvm19 ansible]$ cat ansible.cfg
```

```
[defaults]
inventory  = /home/deyu1/ansible/inventory
roles_path = /home/deyu1/ansible/roles:/usr/share/ansible/roles
```

2. 列出 roles

```
[deyu1@kvm19 ansible]$ ansible-galaxy list
# /home/deyu1/ansible/roles
# /usr/share/ansible/roles
- linux-system-roles.timesync, (unknown version)
- linux-system-roles.selinux, (unknown version)
- linux-system-roles.postfix, (unknown version)
- linux-system-roles.kdump, (unknown version)
- linux-system-roles.network, (unknown version)
- rhel-system-roles.kdump, (unknown version)
- rhel-system-roles.timesync, (unknown version)
- rhel-system-roles.network, (unknown version)
- rhel-system-roles.postfix, (unknown version)
- rhel-system-roles.selinux, (unknown version)
```

9.4 使用 RHEL 角色

1. 查看 timesync 角色文件的範例：

```
[deyu1@kvm19 ansible]$ grep ^Ex -A17 /usr/share/ansible/roles/\
rhel-system-roles.timesync/README.md
Example Playbook
----------------

Install and configure ntp to synchronize the system clock with three NTP
    servers:

```yaml
- hosts: targets
  vars:
    timesync_ntp_servers:
      - hostname: foo.example.com
        iburst: yes
      - hostname: bar.example.com
        iburst: yes
      - hostname: baz.example.com
        iburst: yes
  roles:
    - rhel-system-roles.timesync
```
```

2. 要使用 RHEL 角色，可複製 rhel-system-roles 套件提供的 playbook 範例做修改，不用自己寫。例如：複製 timesync 角色 playbook 範例到 ansible 工作目錄。

```
[deyu1@kvm19 ansible]$ cp /usr/share/doc/rhel-system-roles/timesync/\
example-timesync-playbook.yml timesync.yml
```

3. 編輯複製來的 timesync.yml，所有受管主機使用 timesync 角色啓動校時服務，時間伺服器 192.168.122.1，啓用 iburst 參數。

```
[deyu1@kvm19 ansible]$ vim timesync.yml
[deyu1@kvm19 ansible]$ cat timesync.yml
---
- hosts: all
  vars:
    timesync_ntp_servers:
      - hostname: 192.168.122.1
        iburst: yes
  roles:
    - rhel-system-roles.timesync
```

4. ansible-playbook 執行 timesync.yml 腳本，以下是只做 dev 主機的結果，最後輸出 failed=0 表示成功，過程中有些錯誤是可被忽略 (ignored=6)。

```
[deyu1@kvm19 ansible]$ ansible-playbook timesync.yml

PLAY [dev]
    ********************************************************************


TASK [Gathering Facts]
    *********************************************************
ok: [kvm11]

TASK [rhel-system-roles.timesync : Check if only NTP is needed]
    *****************
ok: [kvm11]

TASK [rhel-system-roles.timesync : Check if single PTP is needed]
    **************
skipping: [kvm11]

TASK [rhel-system-roles.timesync : Check if both NTP and PTP are needed]
     *******
skipping: [kvm11]

```

```
TASK [rhel-system-roles.timesync : Determine current NTP provider]
    *************
ok: [kvm11]

TASK [rhel-system-roles.timesync : Select NTP provider]
    ************************
[DEPRECATION WARNING]: Using tests as filters is deprecated. Instead of
    using
`result|version_compare` use `result is version_compare`. This feature
    will be
removed in version 2.9. Deprecation warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.
ok: [kvm11]

TASK [rhel-system-roles.timesync : Install chrony]
    *****************************
ok: [kvm11]

TASK [rhel-system-roles.timesync : Install ntp]
    *********************************
skipping: [kvm11]

TASK [rhel-system-roles.timesync : Install linuxptp]
    ***************************
skipping: [kvm11]

TASK [rhel-system-roles.timesync : Run phc_ctl on PTP interface]
    ***************
skipping: [kvm11]

TASK [rhel-system-roles.timesync : Check if PTP interface supports HW
    timestamping] ***
skipping: [kvm11]

TASK [rhel-system-roles.timesync : Get chrony version]
    ************************
ok: [kvm11]

TASK [rhel-system-roles.timesync : Get ntp version]
    ****************************
skipping: [kvm11]

TASK [rhel-system-roles.timesync : Generate chrony.conf file]
    ******************
[DEPRECATION WARNING]: Using tests as filters is deprecated. Instead of
    using
`result|version_compare` use `result is version_compare`. This feature
    will be
removed in version 2.9. Deprecation warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.
changed: [kvm11]

TASK [rhel-system-roles.timesync : Generate chronyd sysconfig file]
    ************
```

```
changed: [kvm11]

TASK [rhel-system-roles.timesync : Generate ntp.conf file]
    **********************
skipping: [kvm11]

TASK [rhel-system-roles.timesync : Generate ntpd sysconfig file]
    ***************
skipping: [kvm11]

TASK [rhel-system-roles.timesync : Generate ptp4l.conf file]
    *******************
skipping: [kvm11]

TASK [rhel-system-roles.timesync : Generate ptp4l sysconfig file]
    ***************
skipping: [kvm11]

TASK [rhel-system-roles.timesync : Generate phc2sys sysconfig file]
    *************
skipping: [kvm11]

TASK [rhel-system-roles.timesync : Generate timemaster.conf file]
    ***************
skipping: [kvm11]

TASK [rhel-system-roles.timesync : Update network sysconfig file]
    ***************
changed: [kvm11]

TASK [rhel-system-roles.timesync : Disable chronyd]
    ****************************
skipping: [kvm11]

TASK [rhel-system-roles.timesync : Disable ntpd]
    *******************************
fatal: [kvm11]: FAILED! => {"changed": false, "msg": "Could not find the
    requested service ntpd: host"}
...ignoring

TASK [rhel-system-roles.timesync : Disable ntpdate]
    ****************************
fatal: [kvm11]: FAILED! => {"changed": false, "msg": "Could not find the
    requested service ntpdate: host"}
...ignoring

TASK [rhel-system-roles.timesync : Disable sntp]
    *******************************
fatal: [kvm11]: FAILED! => {"changed": false, "msg": "Could not find the
    requested service sntp: host"}
...ignoring

TASK [rhel-system-roles.timesync : Disable ptp4l]
    *****************************
```

```
fatal: [kvm11]: FAILED! => {"changed": false, "msg": "Could not find the
    requested service ptp4l: host"}
...ignoring

TASK [rhel-system-roles.timesync : Disable phc2sys]
    ***************************
fatal: [kvm11]: FAILED! => {"changed": false, "msg": "Could not find the
    requested service phc2sys: host"}
...ignoring

TASK [rhel-system-roles.timesync : Disable timemaster]
    *************************
fatal: [kvm11]: FAILED! => {"changed": false, "msg": "Could not find the
    requested service timemaster: host"}
...ignoring

TASK [rhel-system-roles.timesync : Enable chronyd]
    ****************************
ok: [kvm11]

TASK [rhel-system-roles.timesync : Enable ntpd]
    *******************************
skipping: [kvm11]

TASK [rhel-system-roles.timesync : Enable ptp4l]
    *******************************
skipping: [kvm11]

TASK [rhel-system-roles.timesync : Enable phc2sys]
    ****************************
skipping: [kvm11]

TASK [rhel-system-roles.timesync : Enable timemaster]
    *************************
skipping: [kvm11]

RUNNING HANDLER [rhel-system-roles.timesync : restart chronyd]
    *****************
changed: [kvm11]

PLAY RECAP
    *********************************************************************

kvm11 : ok=17 changed=4 unreachable=0 failed=0 skipped=18 rescued=0
    ignored=6
```

5. 登入 kvm11 查看 chrony.conf 是否依 timesync.yml 腳本配置校時伺服器及參數？

```
[deyu1@kvm19 ansible]$ ssh kvm11 "grep ^server /etc/chrony.conf"
server 192.168.122.1 iburst
```

9.5 實機練習

1. 安裝 rhel-system-roles 套件。
2. 確認目前工作目錄下的 ansible.cfg，roles_path 是否包含 rhel 系統角色目錄？
3. 撰寫 timesync.yml，在所有受管主機使用 timesync 角色啓用校時服務，時間伺服器 192.168.122.1，啓用 iburst 參數。
4. 撰寫 postfix.yml，對 prod 主機使用 rhel 系統角色 postfix，設定 relay_host 爲 deyu.wang。

Chapter 10

Ansible Galaxy

10.1 前言

1. Ansible Galaxy 是一個共享和下載 Ansible 角色的網站。
2. Ansible Galaxy 網站可免費查詢下載角色，評論各種社群開發的 Ansible 角色。

10.2 ansible-galaxy 命令

1. ansible-galaxy 輔助說明，第一行的 [delete|import|... 是可用的子命令。

```
[deyu1@kvm19 ansible]$ ansible-galaxy --help
Usage: ansible-galaxy [delete|import|info|init|install|list|login|remove
    |
search|setup] [--help] [options] ...

Perform various Role related operations.

Options:
  -h, --help            show this help message and exit
  -c, --ignore-certs    Ignore SSL certificate validation errors.
  -s API_SERVER, --server=API_SERVER
                        The API server destination
  -v, --verbose         verbose mode (-vvv for more, -vvvv to enable
                        connection debugging)
  --version             show program's version number, config file
     location,
                        configured module search path, module location,
                        executable location and exit

 See 'ansible-galaxy <command> --help' for more information on a
     specific
command.
```

2. ansible-galaxy 每個子命令都有輔助說明，查詢 search 的輔助說明。

```
[deyu1@kvm19 ansible]$ ansible-galaxy search --help
Usage: ansible-galaxy search [searchterm1 searchterm2] [--galaxy-tags
    galaxy_tag1,
galaxy_tag2] [--platforms platform1,platform2] [--author username]

Search the Galaxy database by tags, platforms, author and multiple
    keywords.

Options:
  --author=AUTHOR       GitHub username
  --galaxy-tags=GALAXY_TAGS
                        list of galaxy tags to filter by
  -h, --help            show this help message and exit
  -c, --ignore-certs    Ignore SSL certificate validation errors.
  --platforms=PLATFORMS
                        list of OS platforms to filter by
  -p ROLES_PATH, --roles-path=ROLES_PATH
                        The path to the directory containing your roles.
                             The
                        default is the roles_path configured in your
                        ansible.cfg file (/etc/ansible/roles if not
                        configured)
  -s API_SERVER, --server=API_SERVER
                        The API server destination
  -v, --verbose         verbose mode (-vvv for more, -vvvv to enable
                        connection debugging)
  --version             show program's version number, config file
      location,
                        configured module search path, module location,
                        executable location and exit

 See 'ansible-galaxy <command> --help' for more information on a
     specific
command.
```

10.3 search

1. ansible-galaxy 搜尋 haproxy，找到 186 項

```
[deyu1@kvm19 ansible]$ ansible-galaxy search haproxy
Found 186 roles matching your search:

 Name                                    Description
```

```
5 ----                                  -----------
  aaronpederson.ansible-autodeploy      Simple deployment tool with hooks
7 aaronpederson.newrelic                Application Performance
      Monitoring
  acandid.mariadb                       Install and Configure MariaDB
      10.4
9 AerisCloud.librato                    Install and configure the Librato
       Agent
  alikins.haproxy                       HAProxy installation and
      configuration.
11 andrewrothstein.haproxy              installs HAProxy
  AngelFreak.heartbeat                  Ansible role for heartbeat
13 ansiblebit.haproxy                   Role to setup haproxy.
  automium.haproxy                      Role for deploying HAProxy
15 blurrcat.haproxy                     Installs and configure haproxy
  buluma.ansible-role-haproxy           HAProxy installation and
      configuration.
17 yamb00.haproxy                       Installs and configure haproxy
  ZeroDowntime.haproxy                  Ansible role to install and
      configure haproxy
19 ----                                 -----------
```

10.4 install/remove

1. install 從網站下載角色並安裝。

```
1 [deyu1@kvm19 ansible]$ ansible-galaxy install ansiblebit.haproxy
- downloading role 'haproxy', owned by ansiblebit
3 - downloading role from https://github.com/ansiblebit/haproxy/archive/
    master.tar.bz2
- extracting ansiblebit.haproxy to /home/deyu1/ansible/roles/ansiblebit.
    haproxy
5 - ansiblebit.haproxy (master) was installed successfully
```

2. ansible-galaxy list 列出 ansiblebit.haproxy 角色。

```
1 [deyu1@kvm19 ansible]$ ansible-galaxy list | grep ansiblebit
- ansiblebit.haproxy, master
```

3. 查詢 roles 目錄新增了 ansiblebit.haproxy 目錄，目錄內有角色的各種檔案。

```
[deyu1@kvm19 ansible]$ ll roles/ansiblebit.haproxy/
2 total 44
drwxrwxr-x. 2 deyu1 deyu1 4096 Mar 11 11:00 defaults
```

```
drwxrwxr-x. 2 deyu1 deyu1 4096 Mar 11 11:00 handlers
-rw-rw-r--. 1 deyu1 deyu1 1479 Jun 23  2016 LICENSE
drwxrwxr-x. 2 deyu1 deyu1 4096 Mar 11 11:00 meta
-rw-rw-r--. 1 deyu1 deyu1  298 Jun 23  2016 NOTICE.md
-rw-rw-r--. 1 deyu1 deyu1 2524 Jun 23  2016 README.md
-rw-rw-r--. 1 deyu1 deyu1    0 Jun 23  2016 requirements.yml
drwxrwxr-x. 2 deyu1 deyu1 4096 Mar 11 11:00 tasks
drwxrwxr-x. 3 deyu1 deyu1 4096 Mar 11 11:00 templates
drwxrwxr-x. 7 deyu1 deyu1 4096 Mar 11 11:00 tests
-rw-rw-r--. 1 deyu1 deyu1  369 Jun 23  2016 tox.ini
drwxrwxr-x. 2 deyu1 deyu1 4096 Mar 11 11:00 vars
```

4. remove 移除角色 ansiblebit.haproxy。

```
[deyu1@kvm19 ansible]$ ansible-galaxy remove ansiblebit.haproxy
- successfully removed ansiblebit.haproxy
```

5. ansible-galaxy list 沒有列出 ansiblebit.haproxy 角色。

```
[deyu1@kvm19 ansible]$ ansible-galaxy list | grep ansiblebit
```

6. 查詢 roles 目錄，ansiblebit.haproxy 目錄已被刪除。

```
[deyu1@kvm19 ansible]$ ll roles/
total 4
drwxrwxr-x. 4 deyu1 deyu1 4096 Mar  9 11:11 apache
```

10.5 requirements.yml

1. 除了直接從網站下載安裝角色外，也可同時將多個安裝來源及參數寫成 YAML 格式檔，而且副檔名一定要.yml 或.yaml。

2. 通常 ansible-galaxy 安裝要求檔命名成 requirements.yml，以選項 -r 指定此檔案進行安裝。

```
$ ansible-galaxy install -r requirements.yml
```

3. requirements.yml 屬性：

(a) src：必要屬性，指定安裝角色的來源。

(b) scm：如果 src 是 URL，必須指定 scm 是 git 或 hg。

(c) version：角色的版本。

(d) name：角色的名稱。

4. requirements.yml 範例：

```
# from galaxy
- src: yatesr.timezone

# from GitHub
- src: https://github.com/bennojoy/mysql

# from GitHub, overriding the name and specifying a specific tag
- src: https://github.com/bennojoy/mysql
  version: master
  name: mysql_role

# from a webserver, where the role is packaged in a tar.bz2
- src: https://dywang.csie.cyut.edu.tw/materials/haproxy.tar.bz2
  name: haproxy-role

# from Bitbucket
- src: git+http://bitbucket.org/willthames/git-ansible-galaxy
  version: v1.4

# from Bitbucket, alternative syntax and caveats
- src: http://bitbucket.org/willthames/hg-ansible-galaxy
  scm: hg

# from GitLab or other git-based scm
- src: git@gitlab.company.com:mygroup/ansible-base.git
  scm: git
  version: "0.1" #quoted, so YAML doesn't parse this as a floating-point
      value
```

10.6 requirements.yml 安裝實例

1. 在 ansible 目錄下編輯 requirements.yml，角色來源爲 http 網站的 haproxy.tar.bz2，安裝名稱爲 balancer 的角色。

```
[deyu1@kvm19 ansible]$ vim requirements.yml
---
- src: http://dywang.csie.cyut.edu.tw/materials/haproxy.tar.bz2
  name: balancer

```

```
- src: http://dywang.csie.cyut.edu.tw/materials/phpinfo.tar.bz2
  name: phpinfo
```

2. 使用 requirements.yml 安裝角色。

```
[deyu1@kvm19 ansible]$ ansible-galaxy install -r requirements.yml
- downloading role from http://dywang.csie.cyut.edu.tw/materials/haproxy
    .tar.bz2
- extracting balancer to /home/deyu1/ansible/roles/balancer
- balancer was installed successfully
- downloading role from http://dywang.csie.cyut.edu.tw/materials/phpinfo
    .tar.bz2
- extracting phpinfo to /home/deyu1/ansible/roles/phpinfo
- phpinfo was installed successfully
```

3. ansible-galaxy list 列出 balancer 及 phpinfo 兩個角色。

```
[deyu1@kvm19 ansible]$ ansible-galaxy list | egrep 'balancer|phpinfo'
- balancer, (unknown version)
- phpinfo, (unknown version)
```

4. 查詢 roles 目錄，新增了 balancer 及 phpinfo 兩個角色目錄。

```
[deyu1@kvm19 ansible]$ ll roles/
total 8
drwxrwxr-x. 4 deyu1 deyu1 4096 Mar  9 11:11 apache
drwxrwxr-x. 9 deyu1 deyu1 4096 Mar 11 15:02 balancer
drwxrwxr-x. 8 deyu1 deyu1 4096 Mar 11 15:02 phpinfo
```

10.7 使用 haproxy 角色

1. 在 ansible 目錄下新增 roles.yml，在 webservers 群組主機使用角色 apache，安裝 httpd 網頁服務，另在 balancers 群組主機使用角色 balancer 安裝負載平衡代理伺服器 haproxy。

```
[deyu1@kvm19 ansible]$ cat roles.yml
---
- name: install httpd
  hosts: webservers
  roles:
```

```
  - apache

- name: use balancer
  hosts: balancers
  roles:
  - balancer
```

2. ansible-playbook 執行腳本 roles.yml。

```
[deyu1@kvm19 ansible]$ ansible-playbook roles.yml

PLAY [install httpd]
    ************************************************************

TASK [Gathering Facts]
    ************************************************************
ok: [kvm14]
ok: [kvm13]

TASK [apache : install the latest version of Apache]
    ***************************
ok: [kvm14]
ok: [kvm13]

TASK [apache : Start service httpd]
    *****************************************
ok: [kvm14]
ok: [kvm13]

TASK [apache : firewalld]
    ******************************************************
ok: [kvm13]
ok: [kvm14]

TASK [apache : Template a file]
    ************************************************
ok: [kvm13]
ok: [kvm14]

PLAY [use balancer]
    ************************************************************

TASK [Gathering Facts]
    ************************************************************
ok: [kvm15]

TASK [haproxy : Download and install haproxy]
    *********************************
changed: [kvm15]

TASK [haproxy : Configure the haproxy cnf file with hosts]
    *********************
```

```
changed: [kvm15]

TASK [haproxy : Start the haproxy service]
    ***********************************
changed: [kvm15]

RUNNING HANDLER [haproxy : restart haproxy]
    ***********************************
changed: [kvm15]

PLAY RECAP
    *********************************************************************

kvm13 : ok=5 changed=0 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
kvm14 : ok=5 changed=0 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
kvm15 : ok=5 changed=4 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

3. kvm15 的 selinux 沒處理，還不能連線。

```
[deyu1@kvm19 ansible]$ curl http://kvm15/
curl: (7) Failed to connect to kvm15 port 80: No route to host
```

4. 登入 kvm15 測試連線負載平衡器主機 kvm15，第一次回應來自 kvm13，再連線一次，回應來自 kvm14，確定 haproxy 工作正常。

```
[root@kvm15 ~]# curl http://kvm15/
Welcome to kvm13.deyu.wang on 192.168.122.13
[root@kvm15 ~]# curl http://kvm15/
Welcome to kvm14.deyu.wang on 192.168.122.14
```

10.8 使用 phpinfo 角色

1. 再編輯 roles.yml，新增任務 hello php，在 webservers 群組主機使用角色 phpinfo。

```
[deyu1@kvm19 ansible]$ cat roles.yml
---
- name: install httpd
  hosts: webservers
  roles:
```

```
6   - apache

8 - name: use balancer
    hosts: balancers
10  roles:
    - balancer
12
- name: hello php
14  hosts: webservers
    roles:
16  - phpinfo
```

2. ansible-playbook 執行 roles.yml 腳本。

```
[deyu1@kvm19 ansible]$ ansible-playbook roles.yml
2
PLAY [install httpd]
    ************************************************************
4
TASK [Gathering Facts]
    ************************************************************
6 ok: [kvm14]
ok: [kvm13]
8
TASK [apache : install the latest version of Apache]
    ***************************
10 ok: [kvm14]
ok: [kvm13]
12
TASK [apache : Start service httpd]
    *****************************************
14 ok: [kvm14]
ok: [kvm13]
16
TASK [apache : firewalld]
    ********************************************************
18 ok: [kvm14]
ok: [kvm13]
20
TASK [apache : Template a file]
    ***********************************************
22 ok: [kvm14]
ok: [kvm13]
24
PLAY [use balancer]
    ************************************************************
26
TASK [Gathering Facts]
    ************************************************************
28 ok: [kvm15]
```

```
TASK [balancer : Download and install haproxy]
    *********************************
ok: [kvm15]

TASK [balancer : Configure the haproxy cnf file with hosts]
    *******************
ok: [kvm15]

TASK [balancer : Start the haproxy service]
    ***********************************
ok: [kvm15]

PLAY [hello php]
    ****************************************************************

TASK [Gathering Facts]
    ************************************************************
ok: [kvm13]
ok: [kvm14]

TASK [phpinfo : generate php info page]
    ***************************************
changed: [kvm13]
changed: [kvm14]

PLAY RECAP
    ********************************************************************

kvm13 : ok=7 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
kvm14 : ok=7 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
kvm15 : ok=4 changed=0 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

3. curl 連線 kvm13 的 hello.php，回應"Hello PHP World from kvm13.deyu.wang" 及 php 訊息。

```
[deyu1@kvm19 ansible]$ curl -s http://kvm13/hello.php
<?php echo 'Hello PHP World from kvm13.deyu.wang'; ?>
<p>
<?php phpinfo(); ?>
```

4. curl 連線 kvm14 的 hello.php，查看回應訊息。

```
[deyu1@kvm19 ansible]$ curl -s http://kvm14/hello.php
<?php echo 'Hello PHP World from kvm14.deyu.wang'; ?>
<p>
<?php phpinfo(); ?>
```

10.9 實機練習

1. 搜尋並安裝 galaxy 角色：
   (a) ansible-galaxy 搜尋 centos+mariadb，並將結果存到 /home/deyu1/ansible/mariadb.txt
   (b) 安裝角色 fi....mariadb (ansible-galaxy 搜尋 centos+mariadb 取得角色完整名稱) 。
2. 在 ansible 目錄下編輯 requirements.yml，角色來源爲 dywang.csie.cyut.edu.tw 網站：
   (a) haproxy.tar.bz2，安裝名稱爲 balancer 的角色。
   (b) phpinfo.tar.bz2，安裝名稱爲 phpinfo 的角色。
3. 撰寫 yaml 腳本 roles.yml，要求如下：
   (a) 在 webservers 群組主機使用角色 apache，安裝 httpd 網頁服務。
   (b) 在 balancers 群組主機使用角色 balancer 安裝負載平衡代理伺服器 haproxy。
   (c) 在 webservers 群組主機使用角色 phpinfo。
4. 撰寫 mariadb.yml
   (a) 不要修改角色 fi....mariadb 的任務，使用這角色在主機 test 安裝 mariadb。
   (b) 安全起見，使用角色安裝 mariadb，會以隨機字串設定 mysql 用户 root 的密碼，並將密碼寫到目標機的某一檔案上，檔名可以從任務目錄上的 user-root.yml 找到，請將檔名 (含路徑) 寫到 mdbpwfile.txt。
5. 角色安裝好的 mariadb server，root 密碼是隨機產生的字串，使用上是安全的，但爲確認你是 mariadb server 管理者，請登入 test 主機，完成以下任務：
   (a) 查看存放 mariadb root 密碼的檔案，這個檔案檔名就是在 kvm19 查到並存在 mdbpwfile.txt，密碼爲何？請將看到的密碼 (不包含其他任何字元) 寫到/root/myroot.txt。
   (b) 使用 mysqladmin 修改 mariadb root 密碼爲 123qwe。

Chapter 11

LVM 檔案系統

11.1 LVM 相關模組

1. parted 分割硬碟，parted 模組文件。

```
[deyu1@kvm19 ansible]$ ansible-doc parted
```

2. filesystem 檔案系統管理，filesystem 模組文件。

```
[deyu1@kvm19 ansible]$ ansible-doc filesystem
```

3. 查詢 LVM 相關模組，其中 lvg 是設置 vg 的模組，lvol 是設置 lv 的模組。

```
[deyu1@kvm19 ansible]$ ansible-doc -l | grep LVM
aix_filesystem          Configure LVM and NFS file systems for AIX
aix_lvg                 Manage LVM volume groups on AIX
aix_lvol                Configure AIX LVM logical volumes
lvg                     Configure LVM volume groups
lvol                    Configure LVM logical volumes
```

4. lvg 邏輯卷冊組管理模組，查看 lvg 模組文件。

```
[deyu1@kvm19 ansible]$ ansible-doc lvg
```

5. lvol 邏輯卷冊管理模組，查看 lvol 模組文件。

```
[deyu1@kvm19 ansible]$ ansible-doc lvol
```

11.2 *parted 模組

1. parted 模組可分割硬碟，查看 parted 模組的範例，其中參數 part_start 分割起點預設是 0% 碟碟最前端，另一參數 part_end 分割終點預設是 100% 硬碟最後端。

```
[deyu1@kvm19 ansible]$ ansible-doc parted | grep LVM -A6
- name: Create a new primary partition for LVM
  parted:
    device: /dev/sdb
    number: 2
    flags: [ lvm ]
    state: present
    part_start: 1GiB
```

2. 建立 partvg.yml 腳本，新增 dev 主機硬碟 /dev/vda3，分割起點是 3800MiB，終點是硬碟最後端。

```
[deyu1@kvm19 ansible]$ vim partvg.yml
[deyu1@kvm19 ansible]$ cat partvg.yml
---
- hosts: dev
  tasks:
  - name: Create a new primary partition for LVM
    parted:
      device: /dev/vda
      number: 3
      flags: [ lvm ]
      state: present
      part_start: 3800MiB
```

3. 分割碟碟必須先清楚主機硬碟的現況，包含硬碟編號、分割數、分割位置等。ansible-playbook 執行 partvg.yml 腳本，出現無法分割訊息，其中說明指定的分割起點是 3800MiB，但硬碟只能使用 3994624kiB 到 4194304kiB。

```
[deyu1@kvm19 ansible]$ ansible-playbook partvg.yml

PLAY [dev]
    ********************************************************************
```

```
4
TASK [Gathering Facts]
    ***********************************************************
6 ok: [kvm11]

8 TASK [Create a new primary partition for LVM]
    *********************************
fatal: [kvm11]: FAILED! => {"changed": false, "err": "Error: You
    requested a
10 partition from 3891200kiB to 4194304kiB (sectors 7782400..8388607).\nThe
     closest
 location we can manage is 3994624kiB to 4194304kiB (sectors
     7989248..8388607).
12 \n", "msg": "Error while running parted script: /usr/sbin/parted -s -m -
    a
optimal /dev/vda -- unit KiB mkpart primary 3800MiB 100%", "out": "", "
    rc": 1}
14
PLAY RECAP
    ***********************************************************************

16 kvm11 : ok=1 changed=0 unreachable=0 failed=1 skipped=0 rescued=0
    ignored=0
```

4. 修改 partvg.yml 腳本的 part_start 爲 3994624KiB，注意 KiB，K 要大寫。

```
[deyu1@kvm19 ansible]$ vim partvg.yml
2 [deyu1@kvm19 ansible]$ grep part_start partvg.yml
      part_start: 3994624KiB
```

5. ansible-playbook 執行 partvg.yml 腳本，成功分割。

```
1 [deyu1@kvm19 ansible]$ ansible-playbook partvg.yml

3 PLAY [dev]
    ***********************************************************************


5 TASK [Gathering Facts]
    ***********************************************************
ok: [kvm11]
7
TASK [Create a new primary partition for LVM]
    *********************************
9 changed: [kvm11]
```

```
PLAY RECAP
    *********************************************************************

kvm11 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

11.3 *lvg 模組

1. lvg 模組可管理邏輯卷冊組 (volume group)，查看 lvg 模組的範例。

```
[deyu1@kvm19 ansible]$ ansible-doc lvg | grep ^EX -A6
EXAMPLES:

- name: Create a volume group on top of /dev/sda1 with physical extent
    size = 32MB
  lvg:
    vg: vg.services
    pvs: /dev/sda1
    pesize: 32
```

2. partvg.yml 腳本新增產生 vg fieldwork 的任務。

```
[deyu1@kvm19 ansible]$ vim partvg.yml
[deyu1@kvm19 ansible]$ cat partvg.yml
---
- hosts: dev
  tasks:
  - name: Create a new primary partition for LVM
    parted:
      device: /dev/vda
      number: 3
      flags: [ lvm ]
      state: present
      part_start: 3994624KiB

  - name: Create VG
    lvg:
      vg: fieldwork
      pvs: /dev/vda3
      pesize: 2
```

3. ansible-playbook 執行 partvg.yml 腳本，成功產生 fieldwork 邏輯卷冊組，其中警告訊息 pesize 欄位格式是字串，建議數字 2 用引號括起來。

```
[deyu1@kvm19 ansible]$ ansible-playbook partvg.yml

PLAY [dev]
    ********************************************************************


TASK [Gathering Facts]
    *********************************************************
ok: [kvm11]

TASK [Create a new primary partition for LVM]
    *********************************
ok: [kvm11]

TASK [Create VG]
    ****************************************************************
 [WARNING]: The value 2 (type int) in a string field was converted to '2
     ' (type
string). If this does not look like what you expect, quote the entire
    value to
ensure it does not change.

changed: [kvm11]

PLAY RECAP
    ********************************************************************

kvm11 : ok=3 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

11.4 lvol 模組

1. 查看 lvol 模組文件，產生 lv 的範例。

```
[deyu1@kvm19 ansible]$ ansible-doc lvol | grep ^EX -A6
EXAMPLES:

- name: Create a logical volume of 512m
  lvol:
    vg: firefly
    lv: test
    size: '512'
```

2. 編輯 lvfs.yml 腳本，在主機 dev 的 fieldwork vg 新增 test lv，大小 100M。

```
[deyu1@kvm19 ansible]$ cat lvfs.yml
```

```
---
- hosts: dev
  tasks:
  - name: Create a logical volume
    lvol:
      vg: fieldwork
      lv: test
      size: '100'
```

3. ansible-playbook 執行 lvfs.yml 腳本，成功產生 test 邏輯卷冊。

```
[deyu1@kvm19 ansible]$ ansible-playbook lvfs.yml

PLAY [dev]
    ********************************************************************



TASK [Gathering Facts]
    ***********************************************************
ok: [kvm11]

TASK [Create a logical volume]
    ************************************************
changed: [kvm11]

PLAY RECAP
    ********************************************************************

kvm11 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

11.5 filesystem 模組

1. 查看 filesystem 模組文件，格式化檔案系統的範例。

```
[deyu1@kvm19 ansible]$ ansible-doc filesystem | grep ^EX -A5
EXAMPLES:

- name: Create a ext2 filesystem on /dev/sdb1
  filesystem:
    fstype: ext2
    dev: /dev/sdb1
```

2. 編輯 lvfs.yml 腳本，新增格式化任務，將 /dev/fieldwork/test 邏輯卷冊格式化爲 ext4 檔案系統。

```
[deyu1@kvm19 ansible]$ vim lvfs.yml
[deyu1@kvm19 ansible]$ cat lvfs.yml
---
- hosts: dev
  tasks:
  - name: Create a logical volume
    lvol:
      vg: fieldwork
      lv: test
      size: '100'

  - name: Create a ext4
    filesystem:
      fstype: ext4
      dev: /dev/fieldwork/test
```

3. ansible-playbook 執行 lvfs.yml 腳本，成功格式化 test 邏輯卷冊。

```
[deyu1@kvm19 ansible]$ ansible-playbook lvfs.yml

PLAY [dev]
    ***********************************************************************


TASK [Gathering Facts]
    *********************************************************
ok: [kvm11]

TASK [Create a logical volume]
    *************************************************
ok: [kvm11]

TASK [Create a ext4]
    ************************************************************
changed: [kvm11]

PLAY RECAP
    ***********************************************************************

kvm11 : ok=3 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

11.6 移除 lv

1. 寫 lvremove.yml 腳本移除先前新增的 lv /dev/fieldwork/test。

```
[deyu1@kvm19 ansible]$ vim lvremove.yml
[deyu1@kvm19 ansible]$ cat lvremove.yml
---
- hosts: dev
  tasks:
  - block:
    - name: Remove a logical volume
      lvol:
        vg: fieldwork
        lv: test
        state: absent
        force: yes
```

2. ansible-playbook 執行 lvremove.yml 腳本，成功刪除 lv test。

```
[deyu1@kvm19 ansible]$ ansible-playbook lvremove.yml

PLAY [dev]
    ********************************************************************


TASK [Gathering Facts]
    ************************************************************
ok: [kvm11]

TASK [Remove a logical volume]
    ***********************************************
changed: [kvm11]

PLAY RECAP
    ********************************************************************

kvm11 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

11.7 實機練習

1. 撰寫 lvmfs1.yml，在 balancers 主機進行以下任務：

   (a) 使用 parted 模組，分割硬碟 /dev/vda3，起點及終點是硬碟所有剩餘空間。

   (b) 使用 lvg 模組在 /dev/vda3 產生名爲 lvm1vg 的 vg，pesize 爲 4M。

   (c) 使用 lvol 模組在 lvm1vg vg，新增一名爲 lvm1lv，大小爲 50M 的 lv，如果已存在則移除再產生。

(d) 使用 filesystem 模組，將 lv /dev/lvm1vg/lvm1lv 格式化爲 ext3。

(e) 做錯或參數變動時，使用模組移除原有分割區、vg, lv。

Chapter 12

Block 錯誤處理

12.1 Block-rescue-always

1. 前面章節 Playbooks 介紹過區塊 (block)，經常配合 when 修件在某些狀況下執行，當 block 執行任務出現錯誤時，執行 rescue 任務，而 always 則不管有無錯誤都執行。

```
- name: Attempt and graceful roll back demo
  block:
    - debug:
        msg: 'I execute normally'
    - name: i force a failure
      command: /bin/false
    - debug:
        msg: 'I never execute, due to the above task failing, :-('
  rescue:
    - debug:
        msg: 'I caught an error'
    - name: i force a failure in middle of recovery! >:-)
      command: /bin/false
    - debug:
        msg: 'I also never execute :-('
  always:
    - debug:
        msg: "This always executes"
```

2. debug 模組的 msg 列印客製化訊息，如果省略則預設列印”Hello World!”，查看 debug 模組範例。

```
[deyu1@kvm19 ansible]$ ansible-doc debug | grep ^EX -A4
EXAMPLES:

# Example that prints the loopback address and gateway for each host
- debug:
```

```
6       msg: System {{ inventory_hostname }} has uuid {{
            ansible_product_uuid }}
```

12.2 變數查詢

1. rescue 是在 block 執行出錯時才執行，判斷是否執行出錯，要先得知執行內容有哪些參數，使用 setup 模組過濾變數字串來查詢。例如：查詢主機 kvm11 的 ansible_lvm 相關參數如如下：

```
[deyu1@kvm19 ansible]$ ansible -m setup -a "filter=ansible_lvm" kvm11
kvm11 | SUCCESS => {
    "ansible_facts": {
        "ansible_lvm": {
            "lvs": {
                "root": {
                    "size_g": "3.42",
                    "vg": "vg_kvm11"
                },
                "swap": {
                    "size_g": "0.12",
                    "vg": "vg_kvm11"
                },
                "test": {
                    "size_g": "0.10",
                    "vg": "fieldwork"
                }
            },
            "pvs": {
                "/dev/vda2": {
                    "free_g": "0.07",
                    "size_g": "3.61",
                    "vg": "vg_kvm11"
                },
                "/dev/vda3": {
                    "free_g": "0.09",
                    "size_g": "0.19",
                    "vg": "fieldwork"
                }
            },
            "vgs": {
                "fieldwork": {
                    "free_g": "0.09",
                    "num_lvs": "1",
                    "num_pvs": "1",
                    "size_g": "0.19"
                },
                "vg_kvm11": {
                    "free_g": "0.07",
                    "num_lvs": "2",
```

```
                    "num_pvs": "1",
                    "size_g": "3.61"
                }
            }
        },
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false
}
```

12.3 實例-正常任務

1. 腳本 lvfs1.yml 使用 block - rescue - always 語法
   (a) when: 限定只有 fieldwork vg 存在時才執行。
   (b) 在 fieldwork 新增一個名爲 test，大小 200M 的 lv
   (c) rescue: 如果無法新增就改新增 100M 的 lv，且印出"Could not create logical volume of that size"。
   (d) always: 新增後將其格式化爲 ext4。

```
[deyu1@kvm19 ansible]$ vim lvfs1.yml
[deyu1@kvm19 ansible]$ cat lvfs1.yml
---
- hosts: dev
  tasks:
  - block:
    - name: Create a logical volume of 200m
      lvol:
        vg: fieldwork
        lv: test
        size: '200'

    rescue:
    - debug:
        msg: Could not create logical volume of that size
    - name: Create a logical volume of 100m
      lvol:
        vg: fieldwork
        lv: test
        size: '100'

    always:
    - name: Create a ext4
      filesystem:
        fstype: ext4
        dev: /dev/fieldwork/test
```

```
27
    when: ansible_lvm.vgs.fieldwork is defined
```

2. ansible-playbook 執行 lvfs1.yml 腳本，正常新增 200M lv test，且成功格式化。

```
[deyu1@kvm19 ansible]$ ansible-playbook lvfs1.yml
2
PLAY [dev]
    ********************************************************************
4
TASK [Gathering Facts]
    ***********************************************************
6 ok: [kvm11]
8 TASK [Create a logical volume of 200m]
    ****************************************
10 changed: [kvm11]

TASK [Create a ext4]
    ************************************************************
12 changed: [kvm11]

14 PLAY RECAP
    ********************************************************************

kvm11 : ok=3 changed=2 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

3. 登入 kvm11 查看 /dev/fieldwork/test 大小爲 200M，且檔案格式爲 ext4。

```
1 [root@kvm11 ~]# lvdisplay /dev/fieldwork/test
  --- Logical volume ---
3 LV Path                /dev/fieldwork/test
  LV Name                test
5 VG Name                fieldwork
  LV UUID                LCXmQs-ZCJb-S5XF-rKWU-ZqB5-1EHD-m5TViN
7 LV Write Access        read/write
  LV Creation host, time kvm11.deyu.wang, 2020-04-01 19:21:13 +0800
9 LV Status              available
  # open                 0
11 LV Size                200.00 MiB
  Current LE             50
13 Segments              1
  Allocation             inherit
15 Read ahead sectors     auto
  - currently set to     8192
17 Block device          253:2
```

```
[root@kvm11 ~]# lsblk -f /dev/fieldwork/test
NAME           FSTYPE LABEL UUID
    MOUNTPOINT
fieldwork-test ext4         f2bbe681-5832-4d4e-8765-9cd0ea1f2bee
```

12.4 實例-rescue

1. 將 lvfs1.yml 腳本的 hosts 改爲 test，並存成 lvfs2.yml。
2. ansible-playbook 執行 lvfs2.yml 腳本，因 kvm12 的 vg 不足 200M，所以無法新增，所以執行 rescue 任務，列印客製化訊息"Could not create logical volume of that size"，並改新增 100M 的 lv。

```
[deyu1@kvm19 ansible]$ ansible-playbook lvfs2.yml

PLAY [test]
    ********************************************************************

TASK [Gathering Facts]
    ***********************************************************
ok: [kvm12]

TASK [Create a logical volume of 200m]
    ****************************************
fatal: [kvm12]: FAILED! => {"changed": false, "err": "  Insufficient
    free space: 25
extents needed, but only 12 available\n", "msg": "Unable to resize test
    to 200m", "rc": 5}

TASK [debug]
    ********************************************************************
ok: [kvm12] => {
    "msg": "Could not create logical volume of that size"
}

TASK [Create a logical volume of 100m]
    ****************************************
changed: [kvm12]

TASK [Create a ext4]
    ***********************************************************
changed: [kvm12]

PLAY RECAP
    ********************************************************************

kvm12 : ok=4 changed=2 unreachable=0 failed=0 skipped=0 rescued=1
    ignored=0
```

3. 登入 kvm12 查看 /dev/fieldwork/test 大小爲 100M，且檔案格式爲 ext4。

```
[root@kvm12 ~]# lvdisplay /dev/fieldwork/test
  --- Logical volume ---
  LV Path                /dev/fieldwork/test
  LV Name                test
  VG Name                fieldwork
  LV UUID                00btWc-eqJi-2Edq-e3EI-OACb-d7eP-jCbLbK
  LV Write Access        read/write
  LV Creation host, time kvm12.deyu.wang, 2020-04-01 20:13:13 +0800
  LV Status              available
  # open                 0
  LV Size                100.00 MiB
  Current LE             25
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:2

[root@kvm12 ~]# lsblk -f /dev/fieldwork/test
NAME           FSTYPE LABEL UUID
   MOUNTPOINT
fieldwork-test ext4         a1ccbef7-ebd6-479c-8168-1dccb65464f5
```

12.5 實例-debug

1. 將 lvfs2.yml 腳本的 hosts 改爲 balancers，並存成 lvfs3.yml，增加 debug 除錯，當 vg fieldwork 不存在時印出客製化訊息"Volume group does not exist"。

```
[deyu1@kvm19 ansible]$ cat lvfs3.yml
---
- hosts: balancers
  tasks:
  ......................
  - debug:
      msg: Volume group does not exist
    when: ansible_lvm.vgs.fieldwork is undefined
```

2. ansible-playbook 執行 lvfs3.yml 腳本，因 kvm15 的 vg fieldwork 不存在，所以執行 debug 任務，列印客製化訊息"Volume group does not exist"。

```
[deyu1@kvm19 ansible]$ ansible-playbook lvfs3.yml

PLAY [balancers] ***************************************************************

TASK [Gathering Facts] *********************************************************
ok: [kvm15]

TASK [Create a logical volume of 200m] *****************************************
skipping: [kvm15]

TASK [Create a ext4] ***********************************************************
skipping: [kvm15]

TASK [debug] *******************************************************************
ok: [kvm15] => {
    "msg": "Volume group does not exist"
}

PLAY RECAP *********************************************************************
kvm15 : ok=2 changed=0 unreachable=0 failed=0 skipped=2 rescued=0 ignored=0
```

12.6 實機練習

1. 使用 lvol 模組，先移除所有主機 VG fieldwork 的 LV。
2. 撰寫 lvfs.yml 腳本，對所有主機完成以下任務：
   (a) when: 限定只有 fieldwork vg 存在時，才執行新增 lv。
   (b) 在 fieldwork 新增一個名爲 data，大小 240M 的 lv。
   (c) rescue: 如果無法新增就改新增 120M 的 lv，且 debug 印出"Could not create LV data of that size"。
   (d) always: 新增後將其格式化爲 ext3。
   (e) when: 限定只有 fieldwork vg 不存在時，才執行 debug 除錯，印出客製化訊息"VG fieldwork does not exist"。

Chapter 13

Shell 模組

13.1 前言

1. Shell 模組可在受管主機執行 shell 命令。
2. Shell 模組預設使用 /bin/sh 執行 shell 命令，但也可以由 executable 參數指定其他的 shell，例如 /bin/bash。

13.2 模組文件

1. 查詢 shell 模組文件。

```
[deyu1@kvm19 ansible]$ ansible-doc shell
```

2. 查詢 shell 模組文件的範例，執行腳本並將輸出導向到檔案。

```
[deyu1@kvm19 ansible]$ ansible-doc shell | grep ^EX -A3
EXAMPLES:

- name: Execute the command in remote shell; stdout goes to the
    specified file
 on the remote.
  shell: somescript.sh >> somelog.txt
```

3. 切換到 somedir 目錄，執行腳本並將輸出導向到檔案。

```
- name: Change the working directory to somedir/ before executing the
    command.
  shell: somescript.sh >> somelog.txt
```

```
  args:
    chdir: somedir/
```

4. 切換到 somedir 目錄，只在 somelog.txt 不存在時，才執行腳本並將輸出導向到檔案 somelog.txt。

```
# You can also use the 'args' form to provide the options.
- name: This command will change the working directory to somedir/ and
    will
 only run when somedir/somelog.txt doesn't exist.
  shell: somescript.sh >> somelog.txt
  args:
    chdir: somedir/
    creates: somelog.txt
```

5. /bin/sh 不支援 cat < /tmp/*txt，參數 executable 指定使用 /bin/bash。

```
- name: Run a command that uses non-posix shell-isms (in this example /
    bin/sh
 doesn't handle redirection and wildcards together but bash does)
  shell: cat < /tmp/*txt
  args:
    executable: /bin/bash
```

6. 使用樣板變數。

```
- name: Run a command using a templated variable (always use quote
    filter to
 avoid injection)
  shell: cat {{ myfile|quote }}
```

7. expect 互動程式。

```
# You can use shell to run other executables to perform actions inline
- name: Run expect to wait for a successful PXE boot via out-of-band
    CIMC
  shell: |
    set timeout 300
    spawn ssh admin@{{ cimc_host }}

    expect "password:"
    send "{{ cimc_password }}\n"

```

```
    expect "\n{{ cimc_name }}"
    send "connect host\n"

    expect "pxeboot.n12"
    send "\n"
    exit 0
  args:
    executable: /usr/bin/expect
  delegate_to: localhost
```

8. 關閉警告訊息。

```
# Disabling warnings
- name: Using curl to connect to a host via SOCKS proxy (unsupported in
    uri).
 Ordinarily this would throw a warning.
  shell: curl --socks5 localhost:9000 http://www.ansible.com
  args:
    warn: no
```

13.3 修改檔案-單一群組

1. 撰寫 shell1.yml 腳本，對 dev 主機的 /etc/issue，修改成特定的字串。

```
[deyu1@kvm19 ansible]$ vim shell1.yml
[deyu1@kvm19 ansible]$ cat shell1.yml
---
- hosts: all
  tasks:
    - shell:
        "echo 'Development {{inventory_hostname}}' > /etc/issue"
      when: '"dev" in group_names'
```

2. ansible-playbook 執行腳本 shell1.yml。

```
[deyu1@kvm19 ansible]$ ansible-playbook shell1.yml

PLAY [all]
    ********************************************************************


TASK [Gathering Facts]
    *********************************************************
ok: [kvm15]
```

```
ok: [kvm11]
ok: [kvm14]
ok: [kvm13]
ok: [kvm12]

TASK [shell]
    ********************************************************************
skipping: [kvm12]
skipping: [kvm15]
skipping: [kvm13]
skipping: [kvm14]
changed: [kvm11]

PLAY RECAP
    ********************************************************************

kvm11 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
kvm12 : ok=1 changed=0 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
kvm13 : ok=1 changed=0 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
kvm14 : ok=1 changed=0 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
kvm15 : ok=1 changed=0 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

3. ansible 使用 shell 模組查看主機 dev 的 /etc/issue 檔案內容，符合要求。

```
[deyu1@kvm19 ansible]$ ansible dev -m shell -a "cat /etc/issue"
kvm11 | CHANGED | rc=0 >>
Development kvm11
```

13.4 修改檔案-不同群組不同內容

1. 修改 shelli1.yml 腳本成 shell2.yml，dev, test, prod 群組主機的 /etc/issue，寫入字串不同。

```
[deyu1@kvm19 ansible]$ vim shell2.yml
[deyu1@kvm19 ansible]$ cat shell2.yml
---
- hosts: all
  tasks:
    - shell:
        "echo 'Development {{inventory_hostname}}' > /etc/issue"
      when: '"dev" in group_names'
```

```
    - shell:
        "echo 'Test {{inventory_hostname}}' > /etc/issue"
      when: '"test" in group_names'
    - shell:
        "echo 'Production {{inventory_hostname}}' > /etc/issue"
      when: '"prod" in group_names'
```

2. ansible-playbook 執行腳本 shell2.yml。

```
[deyu1@kvm19 ansible]$ ansible-playbook shell2.yml

PLAY [all]
    ********************************************************************


TASK [Gathering Facts]
    ************************************************************
ok: [kvm14]
ok: [kvm15]
ok: [kvm11]
ok: [kvm12]
ok: [kvm13]

TASK [shell]
    *******************************************************************
skipping: [kvm12]
skipping: [kvm15]
skipping: [kvm13]
skipping: [kvm14]
changed: [kvm11]

TASK [shell]
    *******************************************************************
skipping: [kvm11]
skipping: [kvm15]
skipping: [kvm13]
skipping: [kvm14]
changed: [kvm12]

TASK [shell]
    *******************************************************************
skipping: [kvm11]
skipping: [kvm12]
skipping: [kvm15]
changed: [kvm13]
changed: [kvm14]

PLAY RECAP
    ********************************************************************

kvm11 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

```
kvm12 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
kvm13 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
kvm14 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
kvm15 : ok=1 changed=0 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

3. ansible 使用 shell 模組查看主機 dev, test, prod 等主機的 /etc/issue 檔案內容，符合要求。

```
[deyu1@kvm19 ansible]$ ansible dev -m shell -a "cat /etc/issue"
kvm11 | CHANGED | rc=0 >>
Development kvm11

[deyu1@kvm19 ansible]$ ansible test -m shell -a "cat /etc/issue"
kvm12 | CHANGED | rc=0 >>
Test kvm12

[deyu1@kvm19 ansible]$ ansible prod -m shell -a "cat /etc/issue"
kvm14 | CHANGED | rc=0 >>
Production kvm14

kvm13 | CHANGED | rc=0 >>
Production kvm13
```

13.5 實機練習

1. 撰寫 YAML 腳本 sh1.yml，對主機 dev 執行任務，要求如下：
   (a) 使用 shell 模組，切換到 dir 目錄，dir 變數由 ansible-playbook 命令列外加參數決定。
   (b) ansible-playbook 執行時變數 dir 用 /usr/local 測試看看。
   (c) 執行 grep ID /etc/os-release，並導向到 os.txt，若 os.txt 存在，則覆蓋其內容。
2. 修改腳本 sh1.yml 爲 sh2.yml，對所有主機使用 shell 模組：
   (a) 切換到 dir 目錄，dir 變數由 ansible-playbook 命令列外加參數決定。
   (b) ansible-playbook 執行時變數 dir 用 /tmp 測試看看。

(c) 執行 grep ID /etc/os-release，並導向到 os.txt，若 os.txt 存在，則覆蓋其內容。

(d) 但只有 /dev/vda3 存在的主機才執行 shell 任務。

(e) 提示：使用變數 ansible_devices.vda.....，變數名稱用 setup 模組查詢。

3. 撰寫 shell1a.yml，對所有主機執行修改任務，要求如下：

(a) 使用 shell 模組，當主機爲 dev 時才將其 /etc/issue 內容修改爲 Development IPV4。

(b) 其中 IPV4 以變數 ansible_default_ipv4..... 取得主機 ipv4 的 IP。

4. 撰寫 shell2a.yml，對所有主機執行修改任務，要求如下：

(a) 使用 shell 模組在 test, prod 群組主機的 /etc/domain，寫入不同字串 XXXX YYYY。

(b) 其中 XXXX 在 dev, test, prod 分別爲 Development, Test, Production。

(c) YYYY 爲網域名稱，使用變數 ansible_domain 取得。

5. 修改腳本 sh2.yml 爲 sh3.yml，對所有主機使用 shell 模組：

(a) 切換到 dir 目錄，dir 變數由 ansible-playbook 命令列外加參數決定。

(b) ansible-playbook 執行時變數 dir 用 /opt 測試看看。

(c) /dev/vda3 存在的主機，執行 shell 任務 grep ID /etc/os-release，並導向到 id.txt。若 id.txt 存在，則覆蓋其內容。

(d) /dev/vda3 不存在的主機，執行 shell 任務 grep ID /etc/profile，並導向到 id.txt。若 id.txt 存在，則覆蓋其內容。

(e) 提示：使用變數 ansible_devices.vda.....，變數名稱用 setup 模組查詢。

Chapter 14

Group 模組

14.1 前言

1. Group 模組可管理群組帳號。
2. group 模組類似 Linux 指令 groupadd, groupdel, groupmod 等群組帳號管理功能。

14.2 模組文件

1. 查詢 group 模組文件。

```
[deyu1@kvm19 ansible]$ ansible-doc group
```

2. 查詢 group 模組文件的範例，新增群組帳號 somegroup。

```
[deyu1@kvm19 ansible]$ ansible-doc group | grep ^EX -A5
EXAMPLES:

- name: Ensure group "somegroup" exists
  group:
    name: somegroup
    state: present
```

14.3 建立群組帳號

1. 撰寫 group.yml 腳本，增加一群組帳號 sharegrp。

```
[deyu1@kvm19 ansible]$ vim group.yml
[deyu1@kvm19 ansible]$ cat group.yml
---
- hosts: dev
  tasks:
  - name: Ensure group "sharegrp" exists
    group:
      name: sharegrp
      state: present
```

2. ansible-playbook 執行腳本 group.yml。

```
[deyu1@kvm19 ansible]$ ansible-playbook group.yml

PLAY [dev]
    ********************************************************************


TASK [Gathering Facts]
    ***********************************************************
ok: [kvm11]

TASK [Ensure group "sharegrp" exists]
    ****************************************
changed: [kvm11]

PLAY RECAP
    ********************************************************************


kvm11 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

3. ansible 使用 shell 模組查看主機 dev 的群組帳號 sharegrp 已產生，gid 是 1001。

```
[deyu1@kvm19 ansible]$ ansible dev -m shell -a "getent group sharegrp"
kvm11 | CHANGED | rc=0 >>
sharegrp:x:1001:
```

14.4 實機練習

1. 撰寫 yaml 腳本 group.yml，要求如下：

(a) 對 dev 主機執行任務。

(b) 使用 group 模組，增加一群組帳號，帳號名稱爲 sharegrp。

2. 撰寫 yaml 腳本 group1.yml，要求如下：

(a) 對 dev 主機執行任務。

(b) 使用 group 模組，增加一群組帳號，帳號名稱爲 webtest。

3. 撰寫 yaml 腳本 group2.yml，要求如下：

(a) 對 test 主機執行任務。

(b) 使用 group 模組，增加一群組帳號，帳號名稱由命令列外加變數 gname 決定。

Chapter 15

File 模組

15.1 前言

1. File 模組可設定或移除檔案屬性、連結或目錄。
2. file 模組可在遠端建立和刪除檔案 (file)、目錄 (directory)、軟連結 (sym-links) 。
3. 類似 Linux 指令 chown, chmod, ln, mkdir 和 touch 等檔案管理功能。

15.2 模組文件

1. 查詢 file 模組文件。

```
[deyu1@kvm19 ansible]$ ansible-doc file
```

2. 查詢 file 模組文件的範例，變更檔案屬性。

```
[deyu1@kvm19 ansible]$ ansible-doc file | grep ^EX -A7
EXAMPLES:

- name: Change file ownership, group and permissions
  file:
    path: /etc/foo.conf
    owner: foo
    group: foo
    mode: '0644'
```

3. 新增屬性 1777 的檔案 /work。

```
- name: Create an insecure file
  file:
    path: /work
    owner: root
    group: root
    mode: '1777'
```

4. 建立軟連結

```
- name: Create a symbolic link
  file:
    src: /file/to/link/to
    dest: /path/to/symlink
    owner: foo
    group: foo
    state: link
```

5. 建立兩個硬連結

```
- name: Create two hard links
  file:
    src: '/tmp/{{ item.src }}'
    dest: '{{ item.dest }}'
    state: link
  loop:
    - { src: x, dest: y }
    - { src: z, dest: k }
```

6. touch 一個檔案，並使用符號方式設定屬性。

```
- name: Touch a file, using symbolic modes to set the permissions (
    equivalent to 0644)
  file:
    path: /etc/foo.conf
    state: touch
    mode: u=rw,g=r,o=r
```

7. touch 一個檔案，並增加或移除屬性。

```
- name: Touch the same file, but add/remove some permissions
  file:
    path: /etc/foo.conf
    state: touch
```

```
5     mode: u+rw,g-wx,o-rwx
```

8. touch 一個檔案，不變更修改及存取的時間。

```
- name: Touch again the same file, but dont change times this makes the
    task idempotent
  file:
    path: /etc/foo.conf
    state: touch
    mode: u+rw,g-wx,o-rwx
    modification_time: preserve
    access_time: preserve
```

9. 如果目錄不存在，則建立目錄。

```
- name: Create a directory if it does not exist
  file:
    path: /etc/some_directory
    state: directory
    mode: '0755'
```

10. 更新修改及存取的時間。

```
- name: Update modification and access time of given file
  file:
    path: /etc/some_file
    state: file
    modification_time: now
    access_time: now
```

11. 設定存取時間。

```
- name: Set access time based on seconds from epoch value
  file:
    path: /etc/another_file
    state: file
    access_time: '{{ "%Y%m%d%H%M.%S" | strftime(stat_var.stat.atime) }}'
```

12. 變更目錄 (含次目錄及檔案) 的擁有者及群組。

```
- name: Recursively change ownership of a directory
  file:
    path: /etc/foo
    state: directory
    recurse: yes
    owner: foo
    group: foo
```

15.3 建立目錄

1. 撰寫 webtest.yml 腳本，建立目錄 /webtest，/webtest 屬於 webtest 群組，屬性 2775 (owner=rwx, group=rwx, other=rx, setGID)。

```
[deyu1@kvm19 ansible]$ vim webtest.yml
[deyu1@kvm19 ansible]$ cat webtest.yml
---
- name: Web test
  hosts: dev
  tasks:
    - file:
        path: /webtest
        state: directory
        group: webtest
        mode: '2775'
```

2. ansible-playbook 執行腳本 webtest.yml。

```
[deyu1@kvm19 ansible]$ ansible-playbook webtest.yml

PLAY [Web test]
    ***************************************************************

TASK [Gathering Facts]
    ***********************************************************
ok: [kvm11]

TASK [file]
    *****************************************************************************
changed: [kvm11]

PLAY RECAP
    *********************************************************************************

kvm11 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

3. ansible 使用 shell 模組查看主機 test 的 /webtest 目錄屬性，完全符合 webtest.yml 腳本的要求。

```
[deyu1@kvm19 ansible]$ ansible dev -m shell -a "ls -ld /webtest"
kvm11 | CHANGED | rc=0 >>
drwxrwsr-x. 2 root webtest 4096 Mar 16 10:17 /webtest
```

15.4 建立軟連結

1. 修改 webtest.yml 腳本，新增一軟連結 /var/www/html/webtest，連結到目錄 /webtest。

```
[deyu1@kvm19 ansible]$ vim webtest.yml
[deyu1@kvm19 ansible]$ cat webtest.yml
---
- name: Web test
  hosts: dev
  tasks:
    - file:
        path: /webtest
        state: directory
        group: webtest
        mode: '2775'

    - file:
        src: /webtest
        dest: /var/www/html/webtest
        state: link
```

2. ansible-playbook 執行腳本 webtest.yml。

```
[deyu1@kvm19 ansible]$ ansible-playbook webtest.yml

PLAY [Web test]
    ****************************************************************

TASK [Gathering Facts]
    ************************************************************
ok: [kvm11]

TASK [file]
    ********************************************************************
```

```
ok: [kvm11]

TASK [file]
    ********************************************************************
changed: [kvm11]

PLAY RECAP
    ********************************************************************

kvm11 : ok=3 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

3. ansible 使用 shell 模組查看主機 test 的 /var/www/html/webtest 目錄屬性，完全符合 webtest.yml 腳本的要求。

```
[deyu1@kvm19 ansible]$ ansible dev -m shell -a "ls -ld /var/www/html/
    webtest"
kvm11 | CHANGED | rc=0 >>
lrwxrwxrwx. 1 root root 8 Mar 16 10:31 /var/www/html/webtest -> /webtest
```

15.5 新增含內容的檔案

1. file 模組的 touch 只能產生空白檔，使用 copy 模組可以複製檔案，或新增有內容的檔案，copy 模組文件的範例如下：

```
[deyu1@kvm19 ansible]$ ansible-doc copy
- name: Copy using inline content
  copy:
    content: '# This file was moved to /etc/other.conf'
    dest: /etc/mine.conf
```

2. 修改 webtest.yml 腳本，在目錄 /webtest 下新增一個內容爲"WebTest on hostname"。

```
[deyu1@kvm19 ansible]$ vim webtest.yml
[deyu1@kvm19 ansible]$ cat webtest.yml
---
- name: Web test
  hosts: dev
  tasks:
    - file:
        path: /webtest
        state: directory
```

```
        group: webtest
        mode: '2775'

    - file:
        src: /webtest
        dest: /var/www/html/webtest
        state: link

    - copy:
        content: "WebTest on {{ ansible_hostname }}"
        dest: /webtest/index.html
```

3. ansible-playbook 執行腳本 webtest.yml。

```
[deyu1@kvm19 ansible]$ ansible-playbook webtest.yml

PLAY [Web test]
    ****************************************************************

TASK [Gathering Facts]
    ************************************************************
ok: [kvm11]

TASK [file]
    ****************************************************************
ok: [kvm11]

TASK [file]
    ****************************************************************
ok: [kvm11]

TASK [copy]
    ****************************************************************
changed: [kvm11]

PLAY RECAP
    ****************************************************************

kvm11 : ok=4 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

4. ansible 使用 shell 模組查看主機 test 的 /webtest/index.html 內容，完全符合 webtest.yml 腳本的要求。

```
[deyu1@kvm19 ansible]$ ansible dev -m shell -a "cat /webtest/index.html"
kvm11 | CHANGED | rc=0 >>
WebTest on kvm11
```

5. curl 連線 http://kvm11/webtest/index.html，回應 403 Forbidden，原因爲 /webtest 目錄的 selinux fcontext 限制存取。

```
[deyu1@kvm19 ansible]$ curl http://kvm11/webtest/index.html
[deyu1@kvm19 ansible]$ curl http://kvm11/webtest/
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /webtest/
on this server.<br />
</p>
</body></html>
```

15.6 實機練習

1. 撰寫 YAML 腳本 file1.yml，對主機 dev 執行任務，要求如下：
   (a) 新增檔案，檔案名稱由命令列外加變數 file1 決定
   (b) file1 擁有者及群組都是 deyu1，屬性 `rw-rw---`。
2. 撰寫 YAML 腳本 dir1.yml，對主機 dev 執行任務，要求如下：
   (a) 新增目錄，目錄名稱由命令列外加變數 dir1 決定
   (b) dir1 所屬群組是 sharegrp，屬性 `rwxr-xr-x`。
3. 撰寫 YAML 腳本 apache-dev.yml，對主機 dev 執行任務：使用 apache 角色，安裝 apache 網頁服務。
4. 撰寫 YAML 腳本 link1.yml，對主機 dev 執行任務，要求如下：
   (a) 新增一軟連結 /var/www/html/web1，連結到目錄 link1。
   (b) link1 目錄由命令列外加變數 link1 決定，且與 dir1.yml 外加變數 dir1 產生的目錄相同。
5. 撰寫 YAML 腳本 copy1.yml，對主機 dev 執行任務，要求如下：
   (a) 目錄 dir1 新增一檔案 home.html，內容爲 Hometest on IP。
   (b) dir1 爲命令列外加變數，IP 使用 ansible 變數取得主機 IP。
6. 撰寫 YAML 腳本 webtest.yml，對主機 dev 執行任務，要求如下：

(a) 新增群組 webtest。

(b) 新增目錄 /webtest，所屬群群組 webtest，屬性 drwxrwsr-x。

(c) 建立連結 /var/www/html/webtest 連結到 /webtest。

(d) 使用 copy 模組，目標檔案爲 /webtest/index.html，其內容爲"WebTest on hostname"，hostname 使用 ansible_hostname 變數取得。

7. 撰寫 YAML 腳本 webtest1.yml，對主機 dev 執行任務，要求如下：

(a) 新增群組 webtest1。

(b) 新增目錄 /webtest1，所屬群群組 webtest1，屬性 drwxr-xr-x。

(c) 建立連結 /var/www/html/webtest1 連結到 /webtest1。

(d) 使用 copy 模組，目標檔案爲 /webtest1/index1.html，其內容爲"WebTest1 on IP，IP 使用 ansible 變數取得主機 IP。

Chapter 16

Sefcontext 模組

16.1 前言

1. sefcontext 模組可管理檔案 selinux context。
2. sefcontext 模組類似 semanage fcontext 命令。

16.2 模組文件

1. 查詢 sefcontext 模組文件。

```
[deyu1@kvm19 ansible]$ ansible-doc sefcontext
```

2. 查詢 sefcontext 模組文件的範例，設定目標目錄 (含次錄) 的 setype 爲 httpd_git_rw_content_t。

```
[deyu1@kvm19 ansible]$ ansible-doc sefcontext | grep ^EX -A7
EXAMPLES:

- name: Allow apache to modify files in /srv/git_repos
  sefcontext:
    target: '/srv/git_repos(/.*)?'
    setype: httpd_git_rw_content_t
    state: present
```

3. 目標目錄 (含次錄) 的 setype，回復成設定值。

```
- name: Apply new SELinux file context to filesystem
  command: restorecon -irv /srv/git_repos
```

4. policycoreutils-python-utils 套件提供管理 selinux fcontext 的命令 semanage，必須確定安裝此套件，所以再查看 yum 模組的安裝範例。

```
[deyu1@kvm19 ansible]$ ansible-doc yum | grep ^EX -A5
EXAMPLES:

- name: install the latest version of Apache
  yum:
    name: httpd
    state: latest
```

16.3 修改 setype

1. 撰寫 sefcontext.yml 腳本，安裝 policycoreutils-python-utils，設定目錄 /webtest 及其次目錄的 fcontext 爲 httpd_sys_content_t，執行命令 restorecon 將 /wevtest 目錄及其次目錄的 fcontext 還原成預設值。

```
[deyu1@kvm19 ansible]$ vim sefcontext.yml
[deyu1@kvm19 ansible]$ cat sefcontext.yml
---
- name: Web test
  hosts: dev
  tasks:
    - yum:
        name: policycoreutils-python-utils
        state: latest

    - sefcontext:
        target: '/webtest(/.*)?'
        setype: httpd_sys_content_t
        state: present

    - command: restorecon -irv /webtest
```

2. ansible-playbook 執行腳本 sefcontext.yml。

```
[deyu1@kvm19 ansible]$ ansible-playbook sefcontext.yml

PLAY [Web test]
    ****************************************************************
```

```
TASK [Gathering Facts]
    ***********************************************************
ok: [kvm11]

TASK [yum]
    *******************************************************************

changed: [kvm11]

TASK [sefcontext]
    ***************************************************************
changed: [kvm11]

TASK [command]
    ***************************************************************
changed: [kvm11]

PLAY RECAP
    *******************************************************************

kvm11 : ok=4 changed=3 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

3. ansible 使用 shell 模組查看主機 test 的 /webtest 目錄，檔案的 index.html 的 fcontext 已改成 httpd_sys_content_t。

```
[deyu1@kvm19 ansible]$ ansible dev -m shell -a "ls -Z /webtest"
kvm11 | CHANGED | rc=0 >>
system_u:object_r:httpd_sys_content_t:s0 index.html
```

4. 使用 curl 連線 kvm11 的 /webtest 目錄，成功回應訊息。

```
[deyu1@kvm19 ansible]$ curl http://kvm11/webtest/
WebTest on kvm11
```

16.4 實機練習

1. 撰寫 sefcontext.yml 腳本，要求如下：

   (a) 安裝 policycoreutils-python-utils

   (b) 設定目錄 /webtest 及其次目錄的 fcontext 爲 httpd_sys_content_t。

(c) 執行命令 restorecon 將 /wevtest 目錄及其次目錄的 fcontext 還原成預設值。

2. 修改 sefcontext.yml 爲 sefcontext1.yml，要求如下：

(a) 將目錄 /public 及其次目錄的 selinux 檔案 context，設定爲 nfs 分享目錄。

(b) (sefcontext type 可使用 seinfo -t 查詢)。

Chapter 17

Lineinfile 模組

17.1 前言

1. lineinfile 模組可配合正規表示法，修改、取代檔案的內容。
2. lineinfile 模組類似 Linux 的 sed 工具，以行爲單位做處理。
3. 如果要變更檔案中相似的多行，可使用 replace 模組。
4. 如果多行文字 (設定起點及終點) 做插入、更新、刪除，可用 blockinfile。

17.2 模組文件

1. 查詢 lineinfile 模組文件。

```
[deyu1@kvm19 ansible]$ ansible-doc lineinfile
```

2. 查詢 lineinfile 模組文件的範例，修改 selinux 設定爲 enforcing。

```
1 [deyu1@kvm19 ansible]$ ansible-doc lineinfile | grep ^EX -A8
  EXAMPLES:
3
  # NOTE: Before 2.3, option 'dest', 'destfile' or 'name' was used instead
      of 'path'
5 - name: Ensure SELinux is set to enforcing mode
    lineinfile:
7     path: /etc/selinux/config
      regexp: '^SELINUX='
9     line: SELINUX=enforcing
```

3. 確認 sudoers 沒有設定 wheel 群組。

```
- name: Make sure group wheel is not in the sudoers configuration
  lineinfile:
    path: /etc/sudoers
    state: absent
    regexp: '^%wheel'
```

4. 設定 /etc/hosts 的屬性。

```
- name: Replace a localhost entry with our own
  lineinfile:
    path: /etc/hosts
    regexp: '^127\.0\.0\.1'
    line: 127.0.0.1 localhost
    owner: root
    group: root
    mode: '0644'
```

5. 確認 httpd 的 port 爲 8080。

```
- name: Ensure the default Apache port is 8080
  lineinfile:
    path: /etc/httpd/conf/httpd.conf
    regexp: '^Listen '
    insertafter: '^#Listen '
    line: Listen 8080
```

6. 確認增加 httpd 的註解行。

```
- name: Ensure we have our own comment added to /etc/services
  lineinfile:
    path: /etc/services
    regexp: '^# port for http'
    insertbefore: '^www.*80/tcp'
    line: '# port for http by default'
```

7. 如果檔案不存在，則新增檔案，加入一行。

```
- name: Add a line to a file if the file does not exist, without passing
    regexp
  lineinfile:
    path: /tmp/testfile
```

```
    line: 192.168.1.99 foo.lab.net foo
    create: yes
```

8. 確認 JBoss 記憶體設定符合要求。

```
# NOTE: Yaml requires escaping backslashes in double quotes but not in
    single quotes
- name: Ensure the JBoss memory settings are exactly as needed
  lineinfile:
    path: /opt/jboss-as/bin/standalone.conf
    regexp: '^(.*)Xms(\\d+)m(.*)$'
    line: '\1Xms${xms}m\3'
    backrefs: yes
```

9. 確認 ADMIN 是 sudoers。

```
# NOTE: Fully quoted because of the ': ' on the line. See the Gotchas in
    the YAML docs.
- name: Validate the sudoers file before saving
  lineinfile:
    path: /etc/sudoers
    state: present
    regexp: '^%ADMIN ALL='
    line: '%ADMIN ALL=(ALL) NOPASSWD: ALL'
    validate: /usr/sbin/visudo -cf %s
```

17.3 Facts 變數查詢

1. ansible 使用 setup 模組可以列出 ansible_facts 變數，增加 filter=.. 參數可以過濾變數。

```
[deyu1@kvm19 ansible]$ ansible dev -m setup -a "filter=ansible_hostname"
kvm11 | SUCCESS => {
    "ansible_facts": {
        "ansible_hostname": "kvm11",
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false
}
```

2. 如果不知變數完整名稱，可列出所有 ansible_facts 變數，使用管線處理過濾字串，例如：memory，得知總記憶體大小的變數爲 ansible_memory_mb.real.total。

```
[deyu1@kvm19 ansible]$ ansible dev -m setup | grep memory -A16
        "ansible_memory_mb": {
            "nocache": {
                "free": 1589,
                "used": 240
            },
            "real": {
                "free": 1341,
                "total": 1829,
                "used": 488
            },
            "swap": {
                "cached": 0,
                "free": 123,
                "total": 123,
                "used": 0
            }
        },
```

3. 總記憶體大小有另一變數 ansible_memtotal_mb。

```
[deyu1@kvm19 ansible]$ ansible dev -m setup -a "filter=
    ansible_memtotal_mb"
kvm11 | SUCCESS => {
    "ansible_facts": {
        "ansible_memtotal_mb": 1829,
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false
}
```

4. 列出所有 ansible_facts 變數，使用管線處理過濾字串 bios，得知 bios 版本變數爲 ansible_bios_version。

```
[deyu1@kvm19 ansible]$ ansible dev -m setup | grep bios
        "ansible_bios_date": "01/01/2007",
        "ansible_bios_version": "0.5.1",
```

5. 如果變數內容太多，管線處理過濾字串不好判斷，可先將整個 ansible_facgts 變數存檔再查。

```
[deyu1@kvm19 ansible]$ ansible dev -m setup > setup.txt
```

6. 例如 ansible_devices 變數有很多資料，vda 大小變數爲 ansible_devices.vda.size。

```
[deyu1@kvm19 ansible]$ grep ansible_devices setup.txt -A160
        "ansible_devices": {
            "dm-0": {
                "holders": [],
                "host": "",
.........................
                "virtual": 1
            },
            "vda": {
                "holders": [],
                "host": "",
                "links": {
                    "ids": [],
                    "labels": [],
                    "masters": [],
                    "uuids": []
                },
                "model": null,
                "partitions": {
                    "vda1": {
                        "holders": [],
.........................
                "sectors": "8388608",
                "sectorsize": "512",
                "size": "4.00 GB",
                "support_discard": "0",
                "vendor": "0x1af4",
                "virtual": 1
            }
        },
```

17.4 修改檔案內容

1. 下載變數格式檔 report.orig。

```
[deyu1@kvm19 ansible]$ wget http://dywang.csie.cyut.edu.tw/materials/
   report.orig
```

2. 查看需要列出報告 report.orig 的變數。

```
# Variables report
HOST=inventoryhostname
MEMORY=memory_in_MB
VDA_SIZE=disk_vda_size
KERNEL=kernel_version
```

3. 撰寫 report.yml 腳本，使用正規表示法搜尋行首爲指定開鍵字的行，等號"="後的字串取代成指定的變數內容。

```
[deyu1@kvm19 ansible]$ vim report.yml
[deyu1@kvm19 ansible]$ cat report.yml
- name: Report variables
  hosts: test
  tasks:
    - copy:
        src: report.orig
        dest: /root/report.txt
    - lineinfile:
        path: /root/report.txt
        regexp: "^{{item.key}}"
        line: "{{item.key}}={{item.value}}"
      loop:
      - { key: "HOST", value: "{{inventory_hostname}}" }
      - { key: "MEMORY", value: "{{ansible_memtotal_mb}}" }
      - { key: "VDA_SIZE", value: "{{ansible_devices.vda.size}}" }
      - { key: "KERNEL", value: "{{ansible_kernel}}" }
```

4. ansible-playbook 執行腳本 report.yml。

```
[deyu1@kvm19 ansible]$ ansible-playbook report.yml

PLAY [Report variables]
    ************************************************************

TASK [Gathering Facts]
    ************************************************************
ok: [kvm12]

TASK [copy]
    ********************************************************************
changed: [kvm12]

TASK [lineinfile]
    ************************************************************
changed: [kvm12] => (item={'key': 'HOST', 'value': 'kvm12'})
changed: [kvm12] => (item={'key': 'MEMORY', 'value': 1829})
changed: [kvm12] => (item={'key': 'VDA_SIZE', 'value': '4.00 GB'})
```

```
15 changed: [kvm12] => (item={'key': 'KERNEL', 'value': '4.18.0-80.el8.
       x86_64'})

17 PLAY RECAP
       *********************************************************************

   kvm12 : ok=3 changed=2 unreachable=0 failed=0 skipped=0 rescued=0
       ignored=0
```

5. ansible 使用 shell 模組查看主機 test 的 /root/report.txt，檔案內容確實已改成變數查詢結果。

```
   [deyu1@kvm19 ansible]$ ansible test -m shell -a "cat /root/report.txt"
2  kvm12 | CHANGED | rc=0 >>
   # Variables report
4  HOST=kvm12
   MEMORY=1829
6  VDA_SIZE=4.00 GB
   KERNEL=4.18.0-80.el8.x86_64
```

17.5 變數不存在

1. 下載變數格式檔 report1.orig。

```
1  [deyu1@kvm19 ansible]$ wget http://dywang.csie.cyut.edu.tw/materials/
       report1.orig
```

2. 查看需要列出報告 report1.orig 的變數。

```
1  # Variables report
   HOST=inventoryhostname
3  BIOS=BIOS_version
   VDA_SIZE=disk_vda_size
5  VDB_SIZE=disk_vdb_size
   OS=os_family
```

3. 撰寫 report1.yml 腳本，因爲 kvm12 沒有 vdb 硬碟，ansible_devices.vdb.size 不會有定義，執行腳本會報錯無法完成。所以 value 設定使用 var|default('NONE')，當變數 var 不在在時，value 設定成預設值'NONE'，其他變數也都設定預設值'NONE'。

```
[deyu1@kvm19 ansible]$ vim report1.yml
[deyu1@kvm19 ansible]$ cat report1.yml
---
- name: Report variables
  hosts: test
  tasks:
    - copy:
        src: report1.orig
        dest: /root/report1.txt
    - lineinfile:
        path: /root/report1.txt
        regexp: "^{{item.key}}"
        line: "{{item.key}}={{item.value}}"
      loop:
        - { key: "HOST", value: "{{inventory_hostname | default('NONE')}}" }
        - { key: "BIOS", value: "{{ansible_bios_version | default('NONE')}}" }
        - { key: "VDA_SIZE", value: "{{ansible_devices.vda.size | default('NONE')}}" }
        - { key: "VDB_SIZE", value: "{{ansible_devices.vdb.size | default('NONE')}}" }
        - { key: "OS", value: "{{ansible_os_family | default('NONE')}}" }
```

4. ansible-playbook 執行腳本 report1.yml。

```
[deyu1@kvm19 ansible]$ ansible-playbook report1.yml

PLAY [Report variables] ***************************************************

TASK [Gathering Facts] ****************************************************
ok: [kvm12]

TASK [copy] ***************************************************************
changed: [kvm12]

TASK [lineinfile] *********************************************************
changed: [kvm12] => (item={'key': 'HOST', 'value': 'kvm12'})
changed: [kvm12] => (item={'key': 'BIOS', 'value': '0.5.1'})
changed: [kvm12] => (item={'key': 'VDA_SIZE', 'value': '4.00 GB'})
changed: [kvm12] => (item={'key': 'VDB_SIZE', 'value': 'NONE'})
changed: [kvm12] => (item={'key': 'OS', 'value': 'RedHat'})

PLAY RECAP ****************************************************************
```

```
19 kvm12 : ok=3 changed=2 unreachable=0 failed=0 skipped=0 rescued=0
       ignored=0
```

5. ansible 使用 shell 模組查看主機 test 的 /root/report1.txt，內容確實已改成變數查詢結果，且 VDB_SIZE=NONE。

```
1 [deyu1@kvm19 ansible]$ ansible test -m shell -a "cat /root/report1.txt"
kvm12 | CHANGED | rc=0 >>
3 # Variables report
HOST=kvm12
5 BIOS=0.5.1
VDA_SIZE=4.00 GB
7 VDB_SIZE=NONE
OS=RedHat
```

17.6 實機練習

1. 撰寫 linf1.yml，對主機 test 完成以下任務。
   (a) 使用 shell 模組，cat /etc/os-release 覆蓋導向到 /etc/dyw-release。
   (b) 修改 /etc/dyw-release 的 HOME_URL 爲" https://dy-wang.csie.cyut.edu.tw" (包含雙引號)。
   (c) 確認 sudoers 沒有設定 wheel 群組。
   (d) 如果檔案 /tmp/linf1 不存在，則產生檔案並新增一行 linf1_test。
2. 使用 setup 模組查詢主機 test 的 ansible_facts 變數，並導向到目錄 /home/deyu1/ansible 下指定檔案。
   (a) facts1.txt：沒有過濾。
   (b) facts2.txt：filter 過濾 ansible_default_ipv4。
   (c) facts3.txt：管線命令，grep 過濾 ipv4 及其後 4 行的結果。
3. 撰寫 report.yml，對 test 主機執行任務，要求如下：
   (a) 下載變數格式檔 report.orig。
   (b) 使用 copy 模組，將下載的 report.orig 複製到遠端的 /root/report.txt。
   (c) 使用 lineinfile 模組，以正規表示法搜尋行首爲指定開鍵字的行，等號"="後的字串取代成指定的變數內容。

(d) 使用 loop 迴圈，針對檔案 /root/report.txt 中"="左邊關鍵字爲 item.key，找到正確的字串爲 item.value。

4. 修改 report.yml 爲

(a) report.a.yml：目標檔案爲 /root/report.a.txt，增加一 lineinfile 任務，在 /root/report.a.txt 的 HOST 前一行增加一項，key 爲 DATE，value 爲`ansible_date_time.date`。

(b) report.b.yml：只保留 /etc/dyw-release 一項，並使用 loop 迴圈修改以下內容，value 都必須用雙引號括起來。

i. key: `HOME_URL`，vaule: `https://dywang.csie.cyut.edu.tw`

ii. key: `ID`，vaule: 變數 `ansible_distribution`

iii. key: `VERSION_ID`，vaule: 變數 `ansible_distribution_version`

5. 撰寫 report1.yml，對 test 主機執行任務，要求如下：

(a) 下載變數格式檔 report1.orig。

(b) 任務如 report.yml，修改 report1.orig 每項 key 的 value。

(c) value 由搜尋變數取得，當搜尋的變數不存在時，item.value 設定成預設值'NONE'。

6. 修改 report1.yml 爲

(a) report1.a.yml：目標檔案爲 /root/report1.a.txt，增加一 lineinfile 任務，在 /root/report1.a.txt 的 HOST 後一行增加一項，key 爲 DATE，value 爲 `ansible_date_time.date`。

(b) report1.b.yml：執行主機改爲 dev 及 test，目標檔案爲 /root/report1.b.txt，增加一 lineinfile 任務，在 /root/report1.b.txt 的 OS 後一行增加一項，key 爲 DATE，value 爲 `ansible_date_time.date`。

Chapter 18

Cron 模組

18.1 前言

1. Cron 模組可設定例行性工作。
2. 類似 Linux 指令 crontab。

18.2 模組文件

1. 查詢 cron 模組文件。

```
[deyu1@kvm19 ansible]$ ansible-doc cron
```

2. 查詢 cron 模組文件的範例，2 點及 5 點執行”ls -alh > /dev/null”。

```
[deyu1@kvm19 ansible]$ ansible-doc cron | grep ^EX -A55
EXAMPLES:

- name: Ensure a job that runs at 2 and 5 exists. Creates an entry like
    "0 5,2 * * ls -alh > /dev/null"
  cron:
    name: "check dirs"
    minute: "0"
    hour: "5,2"
    job: "ls -alh > /dev/null"
```

3. 刪除名爲”an old job” 的例行性工作。

```
- name: 'Ensure an old job is no longer present. Removes any job that is
    prefixed by "#Ansible: an old job" from the crontab'
```

```
  cron:
    name: "an old job"
    state: absent
```

4. 新增一個 cron job，在 reboot 時執行"/some/job.sh"。

```
- name: Creates an entry like "@reboot /some/job.sh"
  cron:
    name: "a job for reboot"
    special_time: reboot
    job: "/some/job.sh"
```

5. 在 crontab 的最上方新增一行"PATH=/opt/bin"。

```
- name: Creates an entry like "PATH=/opt/bin" on top of crontab
  cron:
    name: PATH
    env: yes
    job: /opt/bin
```

6. 在 PATH 宣告行後面新增一行"APP_HOME=/src/app"。

```
- name: Creates an entry like "APP_HOME=/srv/app" and insert it after
    PATH declaration
  cron:
    name: APP_HOME
    env: yes
    job: /srv/app
    insertafter: PATH
```

7. 在 /etc/cron.d 目錄下新增一個例行性工作檔ansible_yum-autoupdate，每星期二 12 點整執行 job。

```
- name: Creates a cron file under /etc/cron.d
  cron:
    name: yum autoupdate
    weekday: 2
    minute: 0
    hour: 12
    user: root
    job: "YUMINTERACTIVE=0 /usr/sbin/yum-autoupdate"
    cron_file: ansible_yum-autoupdate
```

8. 刪除 /etc/cron.d 目錄下的例行性工作檔ansible_yum-autoupdate。

```
- name: Removes a cron file from under /etc/cron.d
  cron:
    name: "yum autoupdate"
    cron_file: ansible_yum-autoupdate
    state: absent
```

9. 刪除 crontab 的環境變數"APP_HOME=/src/app"。

```
- name: Removes "APP_HOME" environment variable from crontab
  cron:
    name: APP_HOME
    env: yes
    state: absent
```

10. 建立一個不安全的 cron。

```
- name: Create an insecure cron
  cron:
    path: /work
    owner: root
    group: root
    mode: '1777'
```

18.3 建立例行性命令

1. 撰寫 cron.yml 腳本，以 deyu1 用户建立例行命令。

```
[deyu1@kvm19 ansible]$ vim cron.yml
[deyu1@kvm19 ansible]$ cat cron.yml
---
- name: cron test
  hosts: test
  tasks:
    - cron:
        name: "create a cron job for deyu1"
        weekday: '2'
        minute: '0'
        hour: '12'
        user: deyu1
        job: 'echo "My CRON_TEST"'
```

2. ansible-playbook 執行腳本 cron.yml。

```
[deyu1@kvm19 ansible]$ ansible-playbook cron.yml

PLAY [cron test] ****************************************************************

TASK [Gathering Facts] **********************************************************
ok: [kvm12]

TASK [cron] *********************************************************************
changed: [kvm12]

PLAY RECAP **********************************************************************
kvm12 : ok=2  changed=1  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
```

3. ansible 使用 shell 模組查看主機 test 用户 deyu1 的例行性命令。

```
[deyu1@kvm19 ansible]$ ansible test -m shell -a "crontab -u deyu1 -l"
kvm12 | CHANGED | rc=0 >>
#Ansible: create a cron job for deyu1
0 12 * * 2 echo "My CRON_TEST"
```

18.4 實機練習

1. 撰寫以下 YAML 腳本，對主機 test 執行任務，建立用户 deyu1 的例行性工作 cronbin (內容查看參數檔)。

   (a) cron1.yml：每 2 小時

   (b) cron2.yml：6 月每天 0-6 點的 12 分

   (c) cron3.yml：每月 1 及 15 日的 10 點 15 分

   (d) cron4.yml：每週日的 13 點每 3 分鐘

   (e) cron5.yml：每天 18 點 25 分

Chapter 19

Vault 加密

19.1 前言

1. 編寫 playbook 時可能涉及一些敏感資料，例如密碼，使用明文方式儲存敏感資料在 playbook 中是不能被接受的。
2. Vault 字面上意義是金庫、保險庫，在 Ansible 是指加密、解密檔案的技術。
3. Ansible 從 v1.5 開始支援此功能，預設使用 AES 加密技術。

19.2 ansible-vault 命令

1. ansible-vault 命令輔助說明，第一行列出的是所有可用的子命令。

```
[deyu1@kvm19 ansible]$ ansible-vault --help
Usage: ansible-vault [create|decrypt|edit|encrypt|encrypt_string|rekey|
    view]
[options] [vaultfile.yml]

encryption/decryption utility for Ansible data files

Options:
  --ask-vault-pass      ask for vault password
  -h, --help            show this help message and exit
  --new-vault-id=NEW_VAULT_ID
                        the new vault identity to use for rekey
  --new-vault-password-file=NEW_VAULT_PASSWORD_FILE
                        new vault password file for rekey
  --vault-id=VAULT_IDS  the vault identity to use
  --vault-password-file=VAULT_PASSWORD_FILES
                        vault password file
  -v, --verbose         verbose mode (-vvv for more, -vvvv to enable
                        connection debugging)
```

```
  --version             show program's version number, config file
      location,
                        configured module search path, module location,
                        executable location and exit

 See 'ansible-vault <command> --help' for more information on a specific
command.
```

19.3 建立加密檔案

1. create 建立加密檔 vfile.yml，輸入密碼後進入 vi 編輯檔案。

```
[deyu1@kvm19 ansible]$ ansible-vault create vfile.yml
New Vault password:
Confirm New Vault password:
```

2. 查看加密檔 vfile.yml，是一串亂數，必須使用 ansible-vault view 輸入密碼才能看到檔案內容。

```
[deyu1@kvm19 ansible]$ cat vfile.yml
$ANSIBLE_VAULT;1.1;AES256
3666353334386634316462376137666466623738323537366631393963636332631613435636439

353533646166366662316334666536343263336634336434 0
    a6632353161323664313461355266375
353438373330613362623932396138613834356662333365636431646432363562626134623236305
0

3666363538626466610
    a6130623538623031646165633939346665636330633531303939633438320
613237313365636634393634366130303162633238656561613039303266343466645
0

[deyu1@kvm19 ansible]$ ansible-vault view vfile.yml
Vault password:
vfile.yml vault test
```

3. edit 輸入密碼後，才可以編輯加密檔 vfile.yml。

```
[deyu1@kvm19 ansible]$ ansible-vault edit vfile.yml
Vault password:
```

4. 再查看加密檔 vfile.yml，是一串亂數，但與之前的內容不一樣，一樣要使用 ansible-vault view 輸入密碼才能查看檔案內容。

```
[deyu1@kvm19 ansible]$ cat vfile.yml
$ANSIBLE_VAULT;1.1;AES256
30646565386333306437616161313261396165386238633033323534373433366266653831623366

333563613261616164343930643034366138656332616661 0
    a346538363039613439373738316530
373833373830623861346636326330643934656633626338346338383935653734366363323139 61

6537316264323030320
    a656233316538343861396565623838396566623664346563383962373837
386330633038363864326631663837393162343738663435346536303838303832 37

[deyu1@kvm19 ansible]$ ansible-vault view vfile.yml
Vault password:
vfile.yml vault test
1111
```

19.4 加解密現有檔案

1. 建立明文檔 vfile1.yml。

```
[deyu1@kvm19 ansible]$ echo 'vault file 1' > vfile1.yml

[deyu1@kvm19 ansible]$ cat vfile1.yml
vault file 1
```

2. encrypt 加密 vfile1.yml，輸入兩次密碼，成功加密。

```
[deyu1@kvm19 ansible]$ ansible-vault encrypt vfile1.yml
New Vault password:
Confirm New Vault password:
Encryption successful
```

3. 查看加密檔 vfile.yml，是一串亂數。

```
[deyu1@kvm19 ansible]$ cat vfile1.yml
$ANSIBLE_VAULT;1.1;AES256
343633356232353662623232323038336236656431313130306266616164636362343031353538 32

633837623637376661353832656437626630306662643533 0
    a663536393138346631386632353033
323832326664613430393761363334643834386264393262373461646338663239313639646561 37
```

```
6 62643237373333234660
      a623139643162333734636235346464323131356632626534373737616633
  6337
```

4. view 輸入密碼後，才可以檢視加密檔 vfile1.yml。

```
1 [deyu1@kvm19 ansible]$ ansible-vault view vfile1.yml
  Vault password:
3 vault file 1
```

5. decrypt 解密 vfile1.yml，輸入密碼，成功解密。

```
1 [deyu1@kvm19 ansible]$ ansible-vault decrypt vfile1.yml
  Vault password:
3 Decryption successful

5 [deyu1@kvm19 ansible]$ cat vfile1.yml
  vault file 1
```

19.5 密碼檔解密 Vault

1. 密碼是敏感資料，不能直接寫在腳本，實作一加密密碼表 pwlist.yml。

```
  [deyu1@kvm19 ansible]$ vim pwlist.yml
2 [deyu1@kvm19 ansible]$ cat pwlist.yml
  devpw: '123123'
4 prodpw: 'qweqwe'
```

2. encrypt 加密 pwlist.yml，輸入密碼 pw4pwlist，成功加密。

```
  [deyu1@kvm19 ansible]$ ansible-vault encrypt pwlist.yml
2 New Vault password:
  Confirm New Vault password:
4 Encryption successful
```

3. 把 encrypt 加密密碼 pw4pwlist，儲存在 passowrd.txt，供後續解密用。

```
[deyu1@kvm19 ansible]$ echo 'pw4pwlist' > password.txt
[deyu1@kvm19 ansible]$ cat password.txt
pw4pwlist
```

4. 使用密碼檔解密 pwlist.yml。

```
[deyu1@kvm19 ansible]$ ansible-vault view pwlist.yml --vault-password-
    file=password.txt
devpw: '123123'
prodpw: 'qweqwe'
```

19.6 rekey 重設密碼

1. 下載 vault 加密檔 rkvfile.yml。

```
[deyu1@kvm19 ansible]$ wget http://dywang.csie.cyut.edu.tw/materials/
    rkvfile.yml
```

2. 已知 rkvfile.yml 的加密密碼是'qweqwe'，使用 ansible-vault view 輸入密碼查看內容。

```
[deyu1@kvm19 ansible]$ ansible-vault view rkvfile.yml
Vault password:
ansible vault test
edit
```

3. 已知 rkvfile.yml 的加密密碼是'qweqwe'，rekey 重設密碼爲'123qwe'。

```
[deyu1@kvm19 ansible]$ ansible-vault rekey rkvfile.yml
Vault password:
New Vault password:
Confirm New Vault password:
Rekey successful
```

4. 使用新密碼解密 rkvfile.yml。

```
[deyu1@kvm19 ansible]$ ansible-vault view rkvfile.yml
Vault password:
ansible vault test
edit
```

19.7 實機練習

1. 共同要求：ansible 工作目錄 /home/deyu1/ansible。
2. create 一個 vault 加密檔 vfile.1.yml
   (a) 內容爲"vfile.1 test 202x-xx-xx"，其中日期爲今天。
   (b) 密碼爲 vpw1。
3. 加密現有檔案：
   (a) 複製 /etc/passwd 到 ansible 工作目錄。
   (b) 使用 ansible-vault 將 passwd 加密，密碼爲 vpw2。
4. 使用密碼檔解密：
   (a) vfile.1.yml 的密碼存到檔案 vfile.1.pw。
   (b) passwd 的密碼存到檔案 passwd.pw。
   (c) 使用密碼檔看是否可以解密？
5. rekey 重設密碼：
   (a) 下載 vault 加密檔 rkvfile.yml。
   (b) 下載 vault 加密檔 rkvfile1.yml。
   (c) 下載 vault 加密檔 rkpwlist.yml。
   (d) 以上三個檔加密密碼都是'qweqwe'，使用 ansible rekey 都重設密碼爲 vpw3。
   (e) 將密碼 vpw3 儲存爲 rkpassword.txt。
   (f) 使用密碼檔 rkpassword.txt 看是否可以解密？

Chapter 20

User 模組

20.1 前言

1. User 模組可管理用户帳號。
2. user 類似 Linux 指令 useradd, userdel, usermod 等帳號管理功能。

20.2 模組文件

1. 查詢 user 模組文件。

```
[deyu1@kvm19 ansible]$ ansible-doc user
```

2. 查詢 user 模組文件的範例，新增帳號 johnd，且指定群組及 uid。

```
[deyu1@kvm19 ansible]$ ansible-doc file | grep ^EX -A8
EXAMPLES:

- name: Add the user 'johnd' with a specific uid and a primary group of
    'admin'
  user:
    name: johnd
    comment: John Doe
    uid: 1040
    group: admin
```

3. 新增帳號 james，且指定群組及 shell。

```
- name: Add the user 'james' with a bash shell, appending the group '
    admins' an>
```

```
  user:
    name: james
    shell: /bin/bash
    groups: admins,developers
    append: yes
```

4. 刪除帳號 johnd。

```
- name: Remove the user 'johnd'
  user:
    name: johnd
    state: absent
    remove: yes
```

5. 新增帳號 jsmith，產生 ssh 金鑰。

```
- name: Create a 2048-bit SSH key for user jsmith in ~jsmith/.ssh/id_rsa
  user:
    name: jsmith
    generate_ssh_key: yes
    ssh_key_bits: 2048
    ssh_key_file: .ssh/id_rsa
```

6. 新增帳號 james18，設定帳户過期時間。

```
- name: Added a consultant whose account you want to expire
  user:
    name: james18
    shell: /bin/zsh
    groups: developers
    expires: 1422403387
```

7. 移除帳號 james18 的過期時間。

```
- name: Starting at Ansible 2.6, modify user, remove expiry time
  user:
    name: james18
    expires: -1
```

20.3 建立用户

1. 撰寫 users.yml 腳本，新增用户 james 且附屬群組 sharegrp。

```
[deyu1@kvm19 ansible]$ vim users.yml
[deyu1@kvm19 ansible]$ cat users.yml
---
- hosts: dev
  tasks:
  - name: Ensure group "sharegrp" exists
    group:
      name: sharegrp
      state: present
  - name: Add user 'james'
    user:
      name: james
      groups: sharegrp
```

2. ansible-playbook 執行腳本 users.yml。

```
[deyu1@kvm19 ansible]$ ansible-playbook users.yml

PLAY [dev]
    ********************************************************************


TASK [Gathering Facts]
    ***********************************************************
ok: [kvm11]

TASK [Ensure group "sharegrp" exists]
    ****************************************
ok: [kvm11]

TASK [Add user 'james']
    ***********************************************************
changed: [kvm11]

PLAY RECAP
    ********************************************************************

kvm12 : ok=3 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

3. ansible 使用 shell 模組查看主機 dev 的帳號 james。

```
[deyu1@kvm19 ansible]$ ansible dev -m shell -a "getent passwd james"
kvm11 | CHANGED | rc=0 >>
```

```
3 james:x:1001:1002::/home/james:/bin/bash
[deyu1@kvm19 ansible]$ ansible dev -m shell -a "id james"
5 kvm11 | CHANGED | rc=0 >>
uid=1001(james) gid=1002(james) groups=1002(james),1001(sharegrp)
```

20.4 設定帳號密碼

1. 修改 users.yml 腳本，增加 password 密碼設定，明文密碼'123qwe' 要經過 password_hash 函式取得雜湊值，否則會將明文'123qwe' 存入 /etc/shadow 中，而無法使用。經過運算的結果要以雙大括號括起來取其結果，而整串文字要用雙引號括起來，如下顯示：

```
[deyu1@kvm19 ansible]$ vim users.yml
2 [deyu1@kvm19 ansible]$ cat users.yml
---
4 - hosts: dev
  tasks:
6   - name: Ensure group "sharegrp" exists
      group:
8       name: sharegrp
        state: present
10  - name: Add user 'james'
      user:
12      name: james
        groups: sharegrp
14      password: "{{ '123qwe' | password_hash('sha512') }}"
```

2. ansible-playbook 執行腳本 users.yml。

```
[deyu1@kvm19 ansible]$ ansible-playbook users.yml
2
PLAY [dev]
    ********************************************************************


4
TASK [Gathering Facts]
    ***********************************************************
6 ok: [kvm11]

8 TASK [Ensure group "sharegrp" exists]
    ******************************************
ok: [kvm11]
10
TASK [Add user 'james']
    ***********************************************************
```

```
12 changed: [kvm11]

14 PLAY RECAP
    *********************************************************************

kvm11 : ok=3 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

3. 使用密碼'123qwe'，成功登入 james@kvm11。

```
1 [deyu1@kvm19 ansible]$ ssh james@kvm11
james@kvm11's password:
3 Last login: Mon Mar 16 20:32:03 2020
```

4. 登出 james@kvm11。

```
1 [james@kvm11 ~]$ exit
logout
3 Connection to kvm11 closed.
```

20.5 Vault 密碼檔

1. 修改 users.yml 腳本爲 users_vault.yml，密碼使用 vault 加密的 pwlist.yml 的密碼表，密碼變數爲 devpw：

```
1 [deyu1@kvm19 ansible]$ vim users_vault.yml
[deyu1@kvm19 ansible]$ cat users_vault.yml
3 ---
- name: Create users with Valut
5   hosts: dev
  vars_files: pwlist.yml
7   tasks:
  - name: Ensure group "sharegrp" exists
9     group:
        name: sharegrp
11      state: present
  - name: Add user 'james'
13    user:
        name: james
15      groups: sharegrp
        password: "{{ devpw | password_hash('sha512') }}"
```

2. ansible-playbook 執行腳本 users_vault.yml，加選項 --ask-vault-pass 手動輸入密碼解密 pwlist.yml。

```
[deyu1@kvm19 ansible]$ ansible-playbook users_vault.yml --ask-vault-pass
Vault password:

PLAY [Create users with Valut]
    ***********************************************

TASK [Gathering Facts]
    ************************************************************
ok: [kvm11]

TASK [Ensure group "sharegrp" exists]
    ****************************************
ok: [kvm11]

TASK [Add user 'james']
    ************************************************************
changed: [kvm11]

PLAY RECAP
    ********************************************************************

kvm11 : ok=3 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

3. ansible-playbook 執行腳本 users_vault.yml，加選項 --vault-password-file 指定金鑰檔解密 pwlist.yml。

```
[deyu1@kvm19 ansible]$ ansible-playbook users_vault.yml --vault-password
    -file password.txt

PLAY [Create users with Valut]
    ***********************************************

TASK [Gathering Facts]
    ************************************************************
ok: [kvm11]

TASK [Ensure group "sharegrp" exists]
    ****************************************
ok: [kvm11]

TASK [Add user 'james']
    ************************************************************
changed: [kvm11]

PLAY RECAP
    ********************************************************************
```

```
kvm11 : ok=3 changed=1 unreachable=0 failed=0 skipped=0 rescued=0
    ignored=0
```

4. 使用密碼'123qwe'，成功登入 james@kvm11。

```
[deyu1@kvm19 ansible]$ ssh james@kvm11
james@kvm11's password:
Last login: Tue Mar 17 08:25:36 2020 from 192.168.122.19
```

5. 登出 james@kvm11。

```
[james@kvm11 ~]$ exit
logout
Connection to kvm11 closed.
```

20.6 建立不同帳號

1. 修改 users_vault.yml 腳本爲 user4host.yml，dev, test 產生的群組及帳號不同於 prod：

```
[deyu1@kvm19 ansible]$ vim user4hosts.yml
[deyu1@kvm19 ansible]$ cat user4hosts.yml
---
- name: Create users with Valut
  hosts: dev test prod
  vars_files: pwlist.yml
  tasks:
  - block:
    - name: Ensure group "devgrp" exists
      group:
        name: devgrp
        state: present
    - name: Add users
      user:
        name: "{{ item }}"
        groups: devgrp
        password: "{{ devpw | password_hash('sha512') }}"
      loop:
        - peter
        - linda
    when: '"dev" in group_names or "test" in group_names'

  - block:
    - name: Ensure group "prodgrp" exists
```

```
25      group:
          name: prodgrp
27        state: present
      - name: Add user 'rita'
29      user:
          name: rita
31        groups: prodgrp
          password: "{{ prodpw | password_hash('sha512') }}"
33    when: '"prod" in group_names'
```

2. ansible-playbook 執行腳本 user4host.yml，加選項 --ask-vault-pass 輸入密碼解密 pwlist.yml。

```
1  [deyu1@kvm19 ansible]$ ansible-playbook user4hosts.yml --ask-vault-pass
   Vault password:
3
   PLAY [Create users with Valut]
       ***********************************************
5
   TASK [Gathering Facts]
       ***********************************************************
7  ok: [kvm11]
   ok: [kvm12]
9  ok: [kvm13]
   ok: [kvm14]
11
   TASK [Ensure group "devgrp" exists]
       ***************************************
13 skipping: [kvm13]
   skipping: [kvm14]
15 changed: [kvm11]
   changed: [kvm12]
17
   TASK [Add user 'peter']
       ***********************************************************
19 skipping: [kvm13]
   skipping: [kvm14]
21 changed: [kvm11]
   changed: [kvm12]
23
   TASK [Ensure group "prodgrp" exists]
       ***************************************
25 skipping: [kvm11]
   skipping: [kvm12]
27 changed: [kvm13]
   changed: [kvm14]
29
   TASK [Add user 'rita']
       ***********************************************************
31 skipping: [kvm11]
   skipping: [kvm12]
```

```
changed: [kvm13]
changed: [kvm14]

PLAY RECAP
    *********************************************************************

kvm11 : ok=3 changed=2 unreachable=0 failed=0 skipped=2 rescued=0
    ignored=0
kvm12 : ok=3 changed=2 unreachable=0 failed=0 skipped=2 rescued=0
    ignored=0
kvm13 : ok=3 changed=2 unreachable=0 failed=0 skipped=2 rescued=0
    ignored=0
kvm14 : ok=3 changed=2 unreachable=0 failed=0 skipped=2 rescued=0
    ignored=0
```

3. ansible-playbook 執行腳本 user4host.yml，加選項 --vault-password-file 指定金鑰檔解密 pwlist.yml。

```
[deyu1@kvm19 ansible]$ ansible-playbook user4hosts.yml --vault-password-
    file=password.txt

PLAY [Create users with Valut]
    ************************************************

TASK [Gathering Facts]
    ************************************************************
ok: [kvm14]
ok: [kvm13]
ok: [kvm11]
ok: [kvm12]

TASK [Ensure group "devgrp" exists]
    ******************************************
skipping: [kvm13]
skipping: [kvm14]
ok: [kvm12]
ok: [kvm11]

TASK [Add user 'peter']
    ************************************************************
skipping: [kvm13]
skipping: [kvm14]
changed: [kvm11]
changed: [kvm12]

TASK [Ensure group "prodgrp" exists]
    ******************************************
skipping: [kvm11]
skipping: [kvm12]
ok: [kvm13]
ok: [kvm14]
```

```
28
TASK [Add user 'rita']
    ************************************************************
30 skipping: [kvm11]
skipping: [kvm12]
32 changed: [kvm13]
changed: [kvm14]
34
PLAY RECAP
    *********************************************************************

36 kvm11 : ok=3 changed=1 unreachable=0 failed=0 skipped=2 rescued=0
    ignored=0
kvm12 : ok=3 changed=1 unreachable=0 failed=0 skipped=2 rescued=0
    ignored=0
38 kvm13 : ok=3 changed=1 unreachable=0 failed=0 skipped=2 rescued=0
    ignored=0
kvm14 : ok=3 changed=1 unreachable=0 failed=0 skipped=2 rescued=0
    ignored=0
```

20.7 測試帳號

1. 使用新增的帳號 peter 登入 kvm11 及 kvm12，輸入密碼'123123'，成功登入，查詢 id 也有附屬群組 devgrp。

```
1 [deyu1@kvm19 ansible]$ ssh peter@kvm11
peter@kvm11's password:
3 Last login: Thu Mar 19 10:31:48 2020 from 192.168.122.19
[peter@kvm11 ~]$ id
5 uid=1002(peter) gid=1004(peter) groups=1004(peter),1003(devgrp)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
7 [peter@kvm11 ~]$ exit
logout
9 Connection to kvm11 closed.

11 [deyu1@kvm19 ansible]$ ssh peter@kvm12
peter@kvm12's password:
13 [peter@kvm12 ~]$ id
uid=1001(peter) gid=1003(peter) groups=1003(peter),1002(devgrp)
15 context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[peter@kvm12 ~]$ exit
17 logout
Connection to kvm12 closed.
```

2. 使用新增的帳號 rita 登入 kvm13 及 kvm14，輸入密碼'qweqwe'，成功登入，查詢 id 也有附屬群組 devgrp。

```
[deyu1@kvm19 ansible]$ ssh rita@kvm13
rita@kvm13's password:
[rita@kvm13 ~]$ id
uid=1001(rita) gid=1002(rita) groups=1002(rita),1001(prodgrp)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[rita@kvm13 ~]$ exit
logout
Connection to kvm13 closed.

[deyu1@kvm19 ansible]$ ssh rita@kvm14
rita@kvm14's password:
[rita@kvm14 ~]$ id
uid=1001(rita) gid=1002(rita) groups=1002(rita),1001(prodgrp)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[rita@kvm14 ~]$ exit
logout
Connection to kvm14 closed.
```

3. 查詢 dev 及 test 主機不存在 rita 帳號。

```
[deyu1@kvm19 ansible]$ ansible dev -m shell -a "id rita"
kvm11 | FAILED | rc=1 >>
id: ‘rita’ : no such usernon-zero return code

[deyu1@kvm19 ansible]$ ansible test -m shell -a "id rita"
kvm12 | FAILED | rc=1 >>
id: ‘rita’ : no such usernon-zero return code
```

4. 查詢 prod 主機不存在 peter 帳號。

```
[deyu1@kvm19 ansible]$ ansible prod -m shell -a "id peter"
kvm13 | FAILED | rc=1 >>
id: ‘peter’ : no such usernon-zero return code

kvm14 | FAILED | rc=1 >>
id: ‘peter’ : no such usernon-zero return code
```

20.8 實機練習

1. 撰寫 user1.yml，對主機 test 完成以下任務。
   (a) 增加 user alex
      i. comment: Alex Chen

ii. shell : /sbin/nologin

iii. uid: 1099

(b) 增加 user cora

i. comment: Cora Wang

ii. shell : /bin/sh

iii. group: wang

(c) 增加 user grace

i. comment: Grace Wang

ii. shell : /bin/sh

iii. group: wang

(d) 增加 user joy

i. comment: Joy Wang

ii. shell : /bin/sh

iii. group: wang

(e) 屬於群組 wang 的用户，使用 loop 迴圈依所列順序新增，用户名稱及註解 (comment) 變數分別爲 name 及 comm。

(f) 用户所屬群組不存在，則自行新增。

2. 修改 user1.yml 爲 user2.yml

(a) 設定群組 wang 的每個 user 的密碼爲：

```
cora: cora.pw
grace: grace.pw
joy: joy.pw
```

(b) loop 迴圈增加一變數 pw，其值爲密碼的 hash 結果。

3. 依以下要求修改 user2.yml，存成 user3.yml。

(a) 將 user2.yml 要設定的用户密碼寫入密碼檔 wang.pw。

(b) user3.yml 改由密碼檔 wang.pw 取得要設定的密碼。

(c) wang.pw 必須使用 ansible-vault 加密，解密密碼爲 123qwe。

4. 修改 user3.yml 爲 user4.yml，處理的主機包含 dev, test 及 prod。

(a) 增加 user mary

i. comment: Mary Lin
ii. shell : /bin/sh
iii. group: lin

(b) 增加 user tina

i. comment: Tina Lin
ii. shell : /bin/sh
iii. group: lin

(c) 密碼檔 wang.pw 增加密碼列表，lin 群組的每個 user 的密碼：

```
mary: mary.pw
tina: tina.pw
```

(d) 執行主機包含 dev, test 及 prod。

(e) 只有主機 dev 及 test 才產生群組 wang 的 user。

(f) 只有主機 prod 才產生群組 lin 的 user。

(g) 所有主機皆新增 user alex。

(h) 用户所屬群組不存在，則自行新增。