

Ansible

Laboratorio di Reti

Filippo Poltronieri
filippo.poltronieri@unife.it

IT Automation

IT Automation

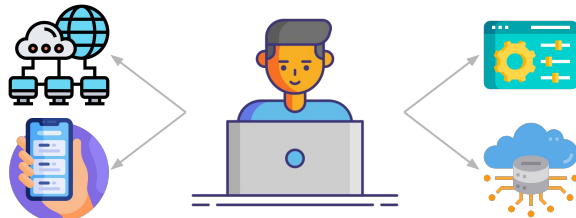
Lavorando nell'IT è probabile che la creazione e la configurazione di infrastrutture IT complesse debbano essere eseguite più volte. Ovviamente risulta molto più comodo automatizzare questi lavori e strumenti di IT automation come Ansible sono utili proprio a questo.



IT Automation: cosa? 1/2

Teoricamente, è possibile automatizzare qualsiasi processo IT, almeno in parte. L'automazione può essere applicata e integrarsi all'automazione della rete, al provisioning di cloud e infrastruttura, agli ambienti operativi standard e perfino al deployment delle applicazioni e alla gestione delle configurazioni.

È possibile estendere l'automazione anche a tecnologie specifiche quali container, a metodologie come DevOps, o ad aree più vaste: cloud, edge computing, sicurezza, test, monitoraggio e avvisi.



IT Automation: cosa? 2/2

- **Provisioning:** di server, utenti, reti e servizi è molto oneroso. Conviene usare dei modelli standard e utilizzare un software che svolga questi compiti automaticamente
- **Gestione delle configurazioni:** le applicazioni richiedono configurazioni sempre più complesse e conviene utilizzare un software automatizzato per memorizzarle e gestirle
- **Orchestrazione:** non solo di container, ma anche di app, servizi e VM, che possono essere distribuiti su cloud privati, ibridi o pubblici
- **Deployment delle applicazioni:** dallo sviluppo al rilascio delle applicazioni ci sono vari passaggi, che conviene automatizzare per ridurre l'errore umano
- **Sicurezza e conformità:** dotarsi di processi e flussi di lavoro standardizzati per garantire la sicurezza significa semplificare la compliance e l'auditing, sapere esattamente come vengono applicati i criteri, e poter verificare che ciò avvenga in modo coerente

IT Automation: quando?

È indicata in qualsiasi scenario, o quasi: un approccio globale all'automazione dell'IT può evitare al personale di dedicarsi a processi ripetitivi e manuali, consentendo una maggiore produttività, riducendo gli errori, migliorando la collaborazione e offrendo più tempo da destinare ad attività maggiormente significative e concettuali.



IT Automation: perché?

Attraverso l'uso dell'IT automation è possibile implementare e gestire qualsiasi servizio in maniera più semplice, rapida e sicura. Gli strumenti di automatizzazione, infatti, non consentono solo di ridurre ai minimi termini il tempo necessario per applicare un'operazione a più dispositivi e sistemi, ma permettono di farlo sfruttando set di impostazioni predefinite, adeguate agli standard fissati a livello di organizzazione.

In altre parole, i processi non sono solo più veloci, ma garantiscono anche un livello di precisione e affidabilità maggiori.



Ansible

Ansible: cos'è?

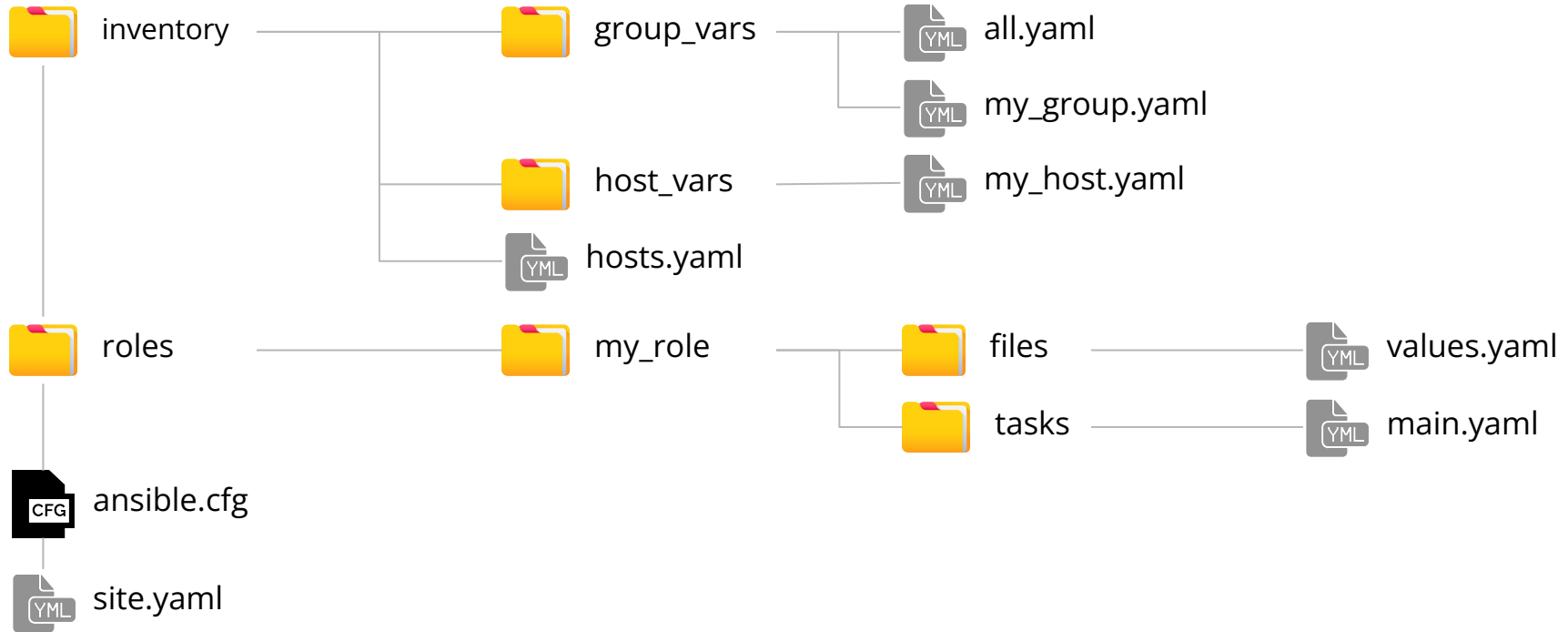


Ansible è un motore di automazione IT open source che automatizza l'installazione, la gestione della configurazione, il deployment delle applicazioni, l'orchestrazione e molti altri processi IT su sistemi UNIX-like e Windows.

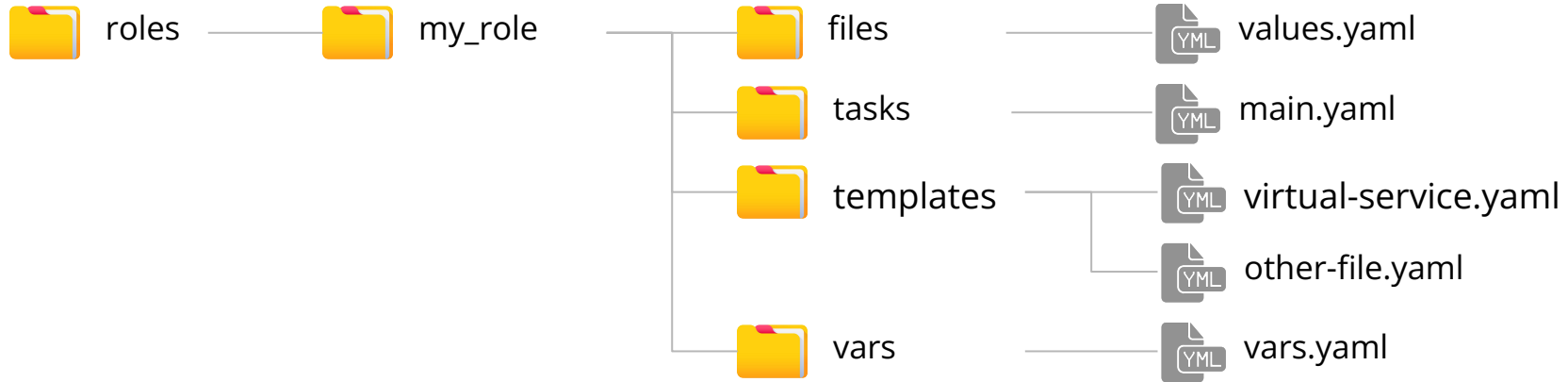
Più precisamente, è un servizio agentless (ovverosia che non richiede degli agent preinstallati nei nodi che gestisce), che si connette tramite SSH a dai nodi, sposta su di essi dei moduli che esegue e poi rimuove al loro termine.

Sebbene sia in grado di configurare macchine con sistemi operativi differenti, **può essere installato e avviato solo su sistemi operativi Unix-like** (Linux, FreeBSD, MacOS, etc..).

Ansible: Struttura progetto



Ansible: Struttura progetto

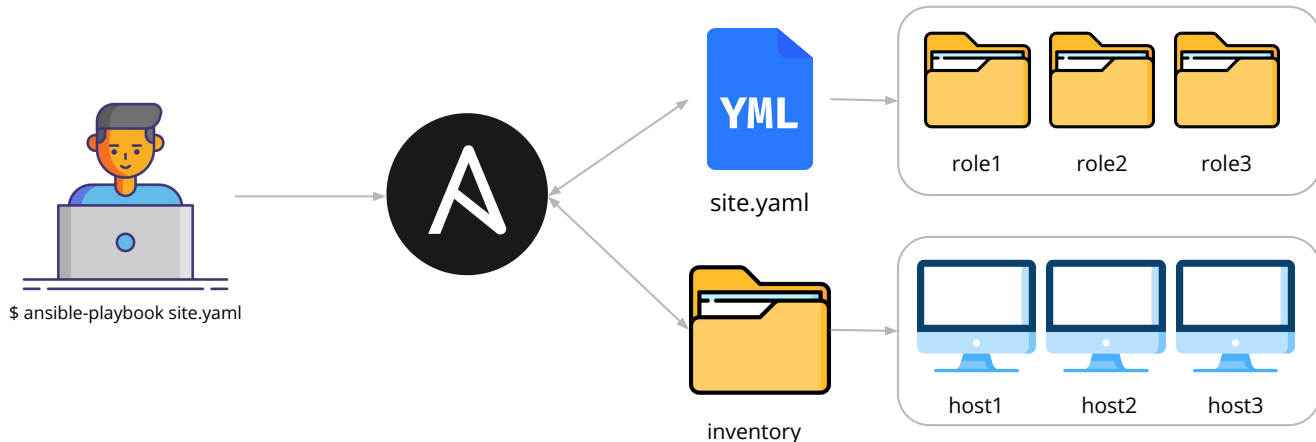


Ansible: Funzionamento 1/2

L'unità minima operativa di Ansible si chiama **playbook**.

Un **file YAML** contenente una serie di attività ordinate, dette *moduli*, che verranno eseguite sequenzialmente sugli host selezionati in un file di inventario Ansible.

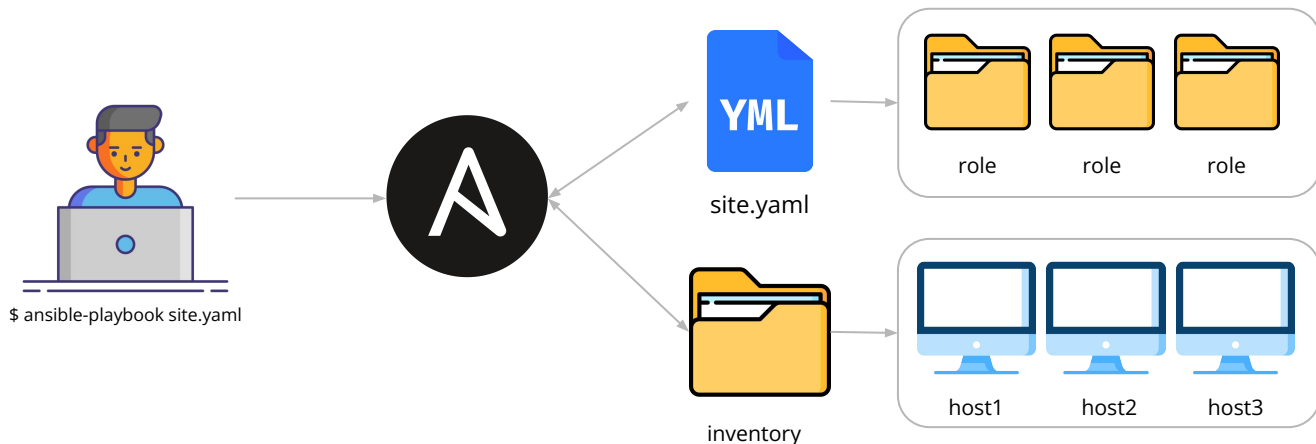
In un progetto deve essere presente l'entry point della playbook, che per comodità si chiamerà `site.yaml`, che dichiara i vari ruoli che verranno eseguiti e con quale utente (normale o root).



Ansible: Funzionamento 2/2

In base agli host configurati in `inventory/hosts.yaml` e alle variabili definite nelle altre sottocartelle dell'inventory, verranno eseguiti i vari ruoli su ciascun host con le rispettive variabili (se presenti).

I ruoli servono a suddividere playbook molto grosse in pezzi più piccoli, dedicati ad esempio a singoli servizi (ad esempio, **nel caso di stack [ELK](#)**, si avrà un ruolo dedicato a Logstash, uno a Elasticsearch e uno a Kibana)



Ansible: File di configurazione

Il file di configurazione di Ansible è utile all'interno di un progetto per definire proprietà specifiche del progetto stesso. Nella maggior parte dei casi, è utile per specificare l'inventory legato al progetto corrente e a saltare la verifica delle chiavi ssh degli host (che andrebbe fatta a mano).

```
# ansible.cfg
[defaults]
inventory = ./inventory
host_key_checking = False
```

Ansible: Inventory

Nella cartella **inventory** è contenuto il file che definisce gli hosts target e i loro gruppi (**hosts.yaml**) e altre cartelle che definiscono le variabili per i singoli host (**host_vars**) o per i gruppi di hosts (**group_vars**).

La cartella **group_vars** contiene dei file che hanno il nome dei gruppi definiti in **hosts.yaml**, oppure dei nomi speciali come ad esempio **all.yaml** che targetizza tutti gli hosts che specificano le variabili utilizzate dai singoli gruppi.

La cartella **host_vars** contiene dei file che hanno il nome dei singoli host definiti in **hosts.yaml** che specificano le variabili utilizzate esclusivamente per i singoli host.

Ansible: Inventory - hosts.yaml, host_vars, group_vars

```
# hosts.yaml
masters:
  hosts:
    my_host:
```

```
# host_vars/my_host.yaml
ansible_host: myhost.com
# ansible_host: 192.168.1.200
ansible_user: myuser
```

```
# group_vars/all.yaml
ansible_ssh_private_key_file: ~/.ssh/private_key
example_var: example_value
```


Ansible: Entrypoint

Entrypoint del progetto, definisce quali sono i ruoli da eseguire, con quale utente eseguirli (utente normale o root, deciso dalla direttiva *become*) e può definire dei tag per raggruppare gruppi di ruoli.

```
# site.yaml
- hosts: my_group
  become: false
  tags:
    - my_tag
  roles:
    - my_role
```

Ansible: Roles

Un ruolo definisce un insieme di operazioni, che per logica devono portare ad un risultato autoconclusivo (ad esempio, l'installazione di un servizio o l'installazione di librerie che possono essere prerequisiti per altri servizi).

Dentro la cartella **roles** vanno definiti i vari ruoli, che non sono altro che altre cartelle con nomi identificativi (ad esempio, se il ruolo installa un server Apache è buona prassi chiamare la cartella *apache*); nell'esempio la cartella si chiama **my_role**.

Ansible: Roles

All'interno dei singoli ruoli deve essere presente la cartella **tasks**, che contiene il file **main.yaml**, che specifica le varie operazioni da eseguire.

Possono poi essere presenti altre cartelle ausiliarie, come ad esempio **files** che generalmente contiene file di configurazione dei servizi che vengono installati e **templates** che invece contiene [template files in formato Jinja2](#) che verranno compilati a runtime sostituendo le variabili con il valore definito nei file dell'inventary (possono essere anche richiesti interattivamente all'utente).

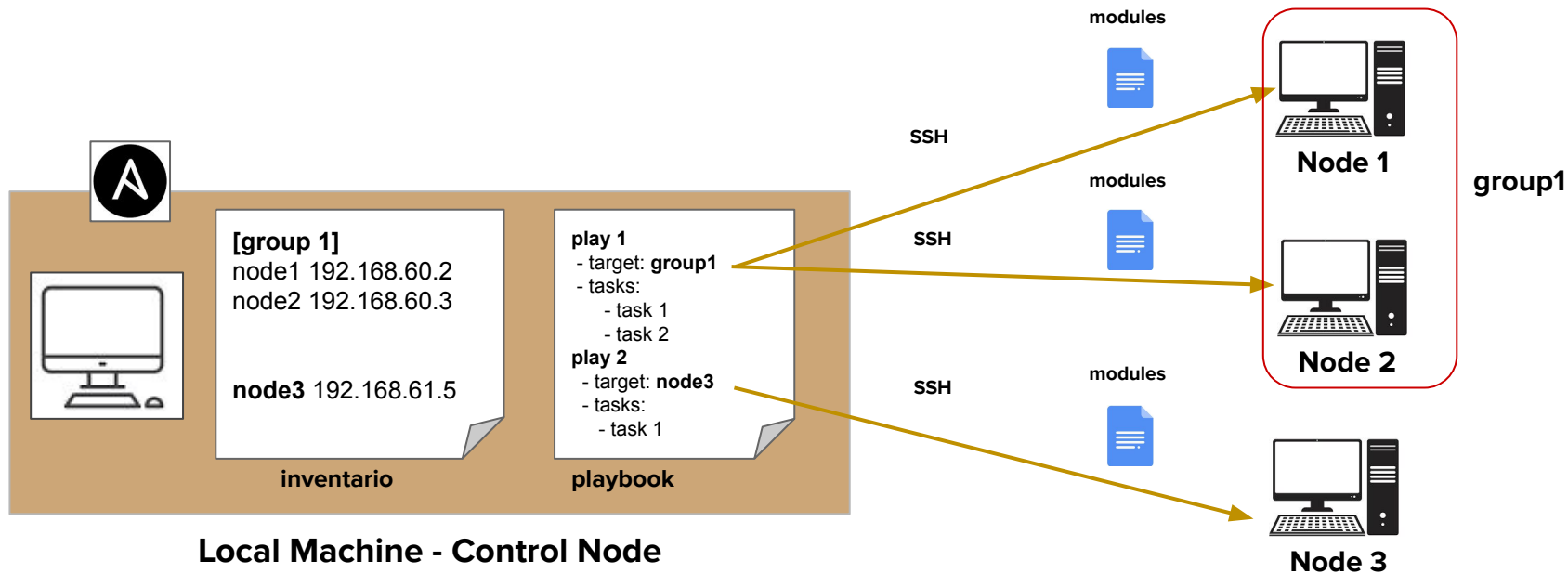
Ansible: Roles - tasks, files, templates

```
# my_role/tasks/main.yaml
- name: Copy file with owner and permissions
  copy:
    src: /srv/myfiles/foo.conf
    dest: /etc/foo.conf
    owner: foo
    group: foo
    mode: 0644
```

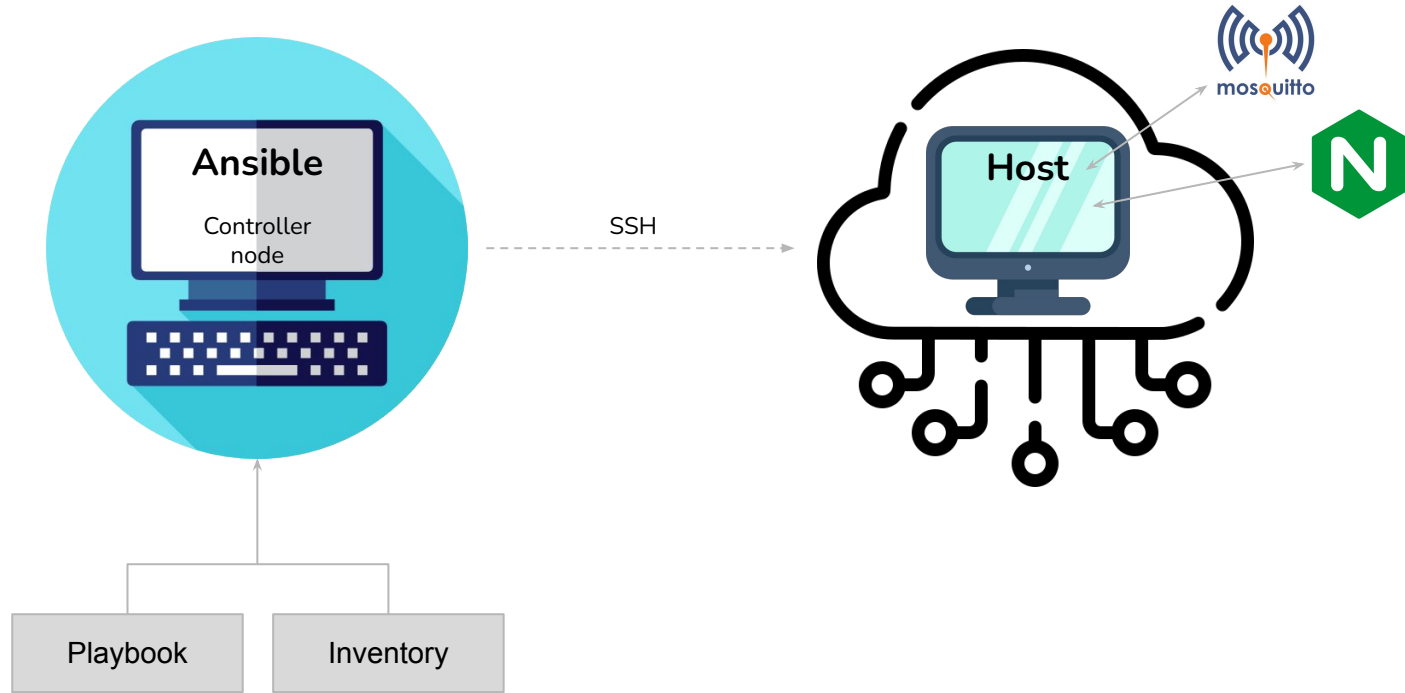
```
# my_role/files/config.cfg
foo=bar
more_foos=more_bars
```

```
# my_role/template/options.ini
foo=bar
more_foos={{ example_var }}
```

Funzionamento



Ansible - Installare servizi su host controllati



Ansible - Installazione

- Ansible è disponibile come pacchetto software nella maggior parte delle distribuzioni Linux (Ubuntu, Debian, Arch Linux).
- Nel caso di Ubuntu:
 - `$ sudo apt update`
 - `$ sudo apt install software-properties-common`
 - `$ sudo apt-add-repository --yes --update ppa:ansible/ansible`
 - `$ sudo apt install ansible`

Ansible - Installazione

- Un'alternativa consiste nell'installare Ansible direttamente tramite il package manager pip
- Nel caso in cui non si abbia ancora installato pip, eseguire il comando:
`sudo apt install python3-pip`
- Eseguire il comando: `pip3 install ansible`
- La documentazione di Ansible è ricchissima e utile:
https://docs.ansible.com/ansible_community.html

Ansible - Setup SSH

- Verificare che sia nel Controller node che negli host sia installato (1) e in funzione (2) il servizio di ssh:

1. `sudo apt list --installed | grep openssh-server`

In caso non sia installato, eseguire il comando:

`sudo apt install openssh-server`

2. `sudo service ssh status`

```
kbuzdar@ubuntu:~$ sudo service ssh status
[sudo] password for kbuzdar:
ssh.service - OpenBSD Secure Shell server
Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
Active: active (running) since Wed 2020-10-07 03:45:38 PDT; 3h 44min ago
Docs: man:sshd(8)
      man:sshd_config(5)
```

In caso non sia in esecuzione, eseguire il comando:

`sudo systemctl start ssh`

Ansible - Setup SSH

Per garantire una corretta comunicazione tra il Controller e l'host, è necessario che quest'ultimo sia in possesso della chiave pubblica SSH del controller.

- Verificare che, sul **Controller**, non sia già presente una coppia di chiavi SSH tramite il comando: `ls -l ~/.ssh/id *.pub`
Se il comando restituisce qualcosa come *"No such file or directory"* o simili, significa che non esistono già chiavi SSH e che è quindi necessario crearle;
- Creare una coppia di chiavi SSH tramite il comando:
`ssh-keygen -t rsa -b 4096`
e premere invio a tutte le successive richieste (per default);

Ansible - Setup SSH

- In caso di successo, l'output del processo dovrebbe essere simile a questo:

```
root@sara-VirtualBox:/etc/ansible# ssh-keygen -t rsa -b 4096 -C "tnnsra@unife.it"
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
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:LZhlyLhCvIw1K208oLee67mcxwC8BLxd7i00dKtvBaI tnnnsra@unife.it
The key's randomart image is:
+---[RSA 4096]-----+
|
|.o   + .
|o B + o o
|oX * o = .
|B+B + o S .
|E=O+ + .
|O.=O+ .
| + Oo
|+#=
+---[SHA256]-----+
```

La configurazione del demone di SSH potrebbe impedire la connessione di un utente root tramite SSH. Cosa fare?

- Verificare se è stato creato, tramite il comando: `ls ~/.ssh/id *`
- Copiare la chiave pubblica del Controller nell'host:
`ssh-copy-id <username-host>@<ip-host>`

Ansible - Setup ambiente

Ora è necessario configurare gli host/inventory (o “managed nodes”) nel controller Node. Per funzionare, Ansible ha bisogno degli indirizzi IP (o nomi logici) dei nodi da controllare;

Esistono diversi metodi per specificare queste informazioni (**/etc/hosts**) o creare un file di inventory che contiene queste informazioni.

Per semplicità utilizzeremo un solo nodo, una macchina virtuale nella rete o anche il proprio host.

Per esempio un nodo managed con indirizzo IP 192.168.1.55

Se non vogliamo utilizzare indirizzi IP statici in locale, un’ottima alternativa è quella di installare il demone [Avahi](#).

Ansible - Setup ambiente

Possiamo inserire le informazioni relative all'host managed in /etc/hosts.

Qui dobbiamo semplicemente includere un entry per associare a un indirizzo IP un nome logico. Dopodiché possiamo utilizzare il nome logico *managed*.

```
#/etc/hosts  
  
# let the system resolves this  
ip  
192.168.1.55 managed
```

Ansible - Setup ambiente

Utilizziamo una configurazione globale per Ansible

```
#/etc/ansible/ansible.cfg

# File configurazione di ansible
[defaults]
inventory=/etc/ansible/inventory.yml
host_key_checking = False
```

Ansible - Setup ambiente

Qui vedremo un setup semplificato rispetto a un progetto Ansible completo.

```
#inventory.yml
web:
  hosts:
    <ip-host>
  vars:
    var_1: red
    var_2: blue
    ansible_user: ubuntu
```

Eeguire i seguenti comandi per generare il file *inventory.yml* del **Controller** node:

- Creare la cartella ansible in /etc se non presente (per impostazione predefinita, Ansible usa questa cartella per archiviare i file di configurazione).
- La configurazione in uso è visualizzabile da terminale digitando il comando:
`$ ansible-config list` (o una delle opzioni del comando)
- Creare e modificare il file di inventory, inserendo le informazioni sugli hosts:
`$ sudo vim /etc/ansible/inventory.yml`

Ansible - Setup ambiente

Per quanto riguarda il file di inventory, possiamo utilizzare la seguente configurazione:

```
#!/etc/ansible/inventory.yml

# Global inventory file for Ansible
web:
  hosts:
    pippo:
      ansible_host: 127.0.0.1
  vars:
    ansible_user: root
```


Ansible - Setup ambiente

Eseguire i seguenti comandi per generare il file *inventory.yml* del **Controller** node:

- Loggarsi come utente root: `sudo su`
- Creare la cartella ansible: `mkdir /etc/ansible`
(per impostazione predefinita, Ansible usa questa cartella per archiviare i file di configurazione)
- Creare e aprire il file di configurazione hosts:
`cat > /etc/ansible/inventory.yml`
- Compilare il file come segue:

```
web:
  hosts:
    <ip-host>
  vars:
    ansible_sudo_pass: <password-host>
    domain: iftsansible.com
    ansible_user: ubuntu
    ansible_password: <password-host>
```

Ansible - Setup ambiente

Dopo aver impostato un file di *inventory.yml*, dobbiamo dire ad Ansible **dove** sono i dettagli dell'Inventory. Per fare ciò, bisogna creare un apposito file di configurazione `/etc/ansible/ansible.cfg`

- Scrivere le seguenti stringhe nel file:

```
[defaults]
```

```
inventory = /etc/ansible/inventory.yml
```

In caso di più inventory? Possiamo utilizzare l'opzione -i quando chiamiamo il comando Ansible.

Ansible - Setup ambiente

- Verificare che Ansible riesca a tracciare i propri host tramite il comando:

```
$ ansible all --list-host
```

Se si riscontrano problemi relativi alla mancanza di sshpass, provare i seguenti:

- Installare sshpass eseguendo il comando:

```
$ sudo apt-get install -y sshpass
```

(Sshpass è un'utility per eseguire SSH in modo non interattivo)

- Verificare la connettività del nodo Controller con l'host tramite il comando:

```
$ ansible all -m ping
```

Ansible - Playbook

Le Ansible Playbook consentono di orchestrare i processi IT.

In particolare una **playbook** è un file YAML contenente una o più play, che permette di definire lo stato desiderato di un sistema.

Un **modulo** Ansible è uno script standalone utilizzabile all'interno di un Ansible Playbook. Esistono moduli builtin e moduli sviluppati dalla community.

Il comando **ansible-galaxy** è l'interfaccia che permette di visualizzare i moduli presenti e di aggiungerne di nuovi.

\$ **ansible-galaxy collection list** # visualizzare i moduli presenti

\$ **ansible-galaxy collection install** <modulo> # installa il <modulo>

Ansible - NGINX (playbook 1)

- Creare la playbook *nginx.yml*, con il seguente contenuto:

```
- hosts: web
  become: yes
  tasks:
    - name: "apt-get update"
      apt:
        update_cache: yes
        cache_valid_time: 3600

    - name: "install nginx"
      apt:
        name: ['nginx']
        state: latest

  handlers:
    - name: restart nginx
      service:
        name: nginx
        state: restarted
```

target

Cosa fa?

nome del modulo

Ansible - NGINX

- Eseguire la playbook per l'installazione e la configurazione di NGINX:
`ansible-playbook nginx.yml`
- Aprire il browser e verificare che il sito funzioni eseguendo:
`http://<ip-host>:80`

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Ansible - Esercizio, come aggiornare un host con apt?

Guardare la documentazione del modulo apt e scrivere una playbook Ansible che consenta di aggiornare tutti i pacchetti installati su uno o più host.

Lanciare la playbook e verificarne il funzionamento.

Ansible - Mosquitto (playbook 1)

- Creare la playbook *mosquitto.yml*, con il seguente contenuto:

```
- hosts: web
  become: yes
  tasks:
    - name: "apt-get update"
      apt:
        update_cache: yes
        cache_valid_time: 3600

    - name: "install mosquitto"
      apt:
        name: ['mosquitto']
        state: latest
```


Ansible - Mosquitto

- Eseguire la playbook per l'installazione di mosquitto:
`ansible-playbook mosquitto.yml`
- Verificare l'avvenuta installazione digitando il seguente comando da terminale dell'host:
`sudo systemctl status mosquitto`
- Testare il funzionamento del broker Mosquitto implementando un publisher.py e un subscriber.py

Ansible - Mosquitto

- Provare il funzionamento del broker con i tool **mosquitto_sub** e **mosquitto_pub**

Ansible - Modifica file di configurazione

- Modificare la configurazione utilizzando il modulo builtin di Ansible **lineinfile**
- **Lineinfile** è un modulo molto utile per fare piccole modifiche a file di configurazione, es. cambiare l'indirizzo e porta di listening;

Esempio, modifica del file /etc/hosts:

- name: Replace a localhost entry searching for a literal string to avoid escaping

```
lineinfile:
  path: /etc/hosts
  search_string: '127.0.0.1'
  line: 127.0.0.1 localhost
  owner: root
  group: root
  mode: '0644'
```

Ansible - Modifica file di configurazione

Un'altro esempio utile è la modifica della porta di listening di un servizio utilizzando una regexp:

- name: Ensure the default Apache port is 8080
 ansible.builtin.lineinfile:
 path: /etc/httpd/conf/httpd.conf
 regexp: '^Listen ' #se è presente Listen
 insertafter: '^#Listen '
 line: Listen 8080

Nella documentazione ufficiale è disponibile un buon numero di esempi e di configurazioni.

Ansible - Esercizio

Proviamo ora a installare Mosquitto, un broker MQTT.

MQTT è un protocollo lightweight publish-subscribe utilzzatissimo nella comunicazione M2M e nel mondo IoT.

Mosquitto è disponibile come pacchetto apt in Ubuntu e Debian. Dobbiamo creare un file di configurazione Ansible per l'installazione e la configurazione di Mosquitto su un nodo di interesse (anche localhost).

- 1) Scrivere il task per l'installazione e l'avvio;
- 2) Verificare lo stato del servizio, è attivo? Su che indirizzo e porta risulta essere in ascolto?
- 3) Scrivere un task Ansible per modificare la configurazione del servizio.

Ansible - NGINX con configurazione

- Effettuare il deployment di una pagina web statica con nginx

```
# site.conf.j2
server {
    listen 80;
    listen [::]:80;
    server_name {{ domain }};
    root /var/www/{{ domain }};
    location / {
        try_files $uri $uri/ =404;
    }
}
```

Possiamo farlo utilizzando il linguaggio di templating jinja. Domain sarà il nome di dominio, il suo valore verrà preso dalla rispettiva variabile.

Ansible - NGINX (playbook 3)

- Creare la playbook *nginx2.yml*, con il seguente contenuto:

```
- hosts: web
  become: yes
  tasks:
    - name: "apt-get update"
      apt:
        update_cache: yes
        cache_valid_time: 3600

    - name: "install nginx"
      apt:
        name: ['nginx']
        state: latest

    - name: "create www directory"
      file:
        path: /var/www/{{ domain }}
        state: directory
        mode: '0775'
        owner: "{{ ansible_user }}"
        group: "{{ ansible_user }}"
```



```
- name: copy nginx site.conf
  template:
    src: site.conf.j2
    dest: /etc/nginx/sites-enabled/{{ domain }}
    owner: root
    group: root
    mode: 0644

handlers:
  - name: restart nginx
    service:
      name: nginx
      state: restarted
```

Ansible - NGINX (Copy files)

- Creare la playbook *copy.yml*, ci servirà per sincronizzare il contenuto di una cartella locale (sul controller) con la cartella sul web server:

```
- hosts: web
  tasks:
    - name: "Copy the content of the website"
      copy:
        src: site/
        dest: /var/www/{{ domain }}
        owner: root
        group: root
        mode: 0644
```


Ansible - NGINX (Sync files)

- Creare la playbook *sync.yml*, ci servirà per sincronizzare il contenuto di una cartella locale (sul controller) con la cartella sul web server:

```
- hosts: web
  tasks:
    - name: "sync website"
      synchronize:
        src: site/
        dest: /var/www/{{ domain }}
        archive: no
        checksum: yes
        recursive: yes
        delete: yes
      become: no
```

Ansible - NGINX

- Ora creiamo la cartella *site* e all'interno di questa una semplicissima pagina HTML, che sarà il nostro nuovo sito web. Per esempio, :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Welcome to labreti.example!</title>
  </head>
  <body>
    <h1>Success! The server block is working!</h1>
  </body>
</html>
```

Ansible - NGINX

- Eseguire la playbook per l'installazione e la configurazione di NGINX:
`ansible-playbook nginx.yml`
- Eseguire la playbook per sincronizzare i file del sito:
`ansible-playbook sync.yml`
- Aprire il browser e verificare che il sito funzioni eseguendo:
`http://<ip-host>:80`

Ansible - Apache Deploy Static Site

- Provare a fare il deploy dello stesso sito utilizzando apache come web server: <https://httpd.apache.org/docs/2.4/vhosts/examples.html>
- Nella playbook inserire anche il task da eseguire per disinstallare nginx che è stato installato negli esercizi precedenti => possibile conflitto sulla porta 80;
 - In alternativa si può pensare di scrivere un task per stoppare e disabilitare nginx con il modulo [service](#) o con il modulo [systemd](#);
- Verificare il funzionamento della playbook utilizzando il browser: aprire il dominio example.com.

Ansible - Modulo systemd

Il parametro state può assumere diversi valori tra cui:

- reloaded (utile per ricaricare la configurazione)
- restarted
- started
- Stopped

“started/stopped are idempotent actions that will not run commands unless necessary. restarted will always bounce the unit. reloaded will always reload.”

Ansible - Modulo systemd

- systemd è parte del core di Ansible e si presenta come un'ottima alternativa a service
- Vediamo qualche esempio su come utilizzarlo

Enable di un servizio, per esempio mosquitto → # systemctl enable mosquitto

```
- name: Enable mosquitto service
```

```
  ansible.builtin.systemd:
```

```
    name: mosquitto
```

```
    enabled: yes
```

posso anche non
specificare
ansible.builtin

specificando **yes** e **no** è
possibile abilitare o disabilitare
un servizio

Ansible - Modulo systemd

Effettuare lo stop di un servizio può essere un'operazione utilissima, quando per esempio vogliamo mandare in esecuzione due servizi che potrebbero essere in conflitto, e.g., nginx e apache.

- name: Stop service nginx

```
ansible.builtin.systemd:
```

```
  name: nginx
```

```
  state: stopped
```

Ansible - Modulo file

Il modulo file è utilissimo per effettuare tutte le operazioni che riguardano la gestione dei [file](#):

- Impostare e cambiare gli attributi di un file
- creare symlinks o directory

Tra i parametri più importanti troviamo src, dest, path, owner, group, mode, state, etc..

Esempio, creazione di un link simbolico → `$ ln -s file/to/link/to /path/to/symlink`

```
- name: Create a symbolic link
  ansible.builtin.file:
    src: /file/to/link/to
    dest: /path/to/symlink
    owner: foo
    group: foo
    state: link
```


Ansible - Modulo file

Un'altro esempio utile è quello di cambiare ricorsivamente i permessi di una directory.

- name: Recursively change ownership of a directory
ansible.builtin.file:
 path: /etc/foo
 state: directory
 recurse: yes
 owner: foo
 group: foo

Ansible - Modulo file

Il parametro **state** può assumere diversi valori:

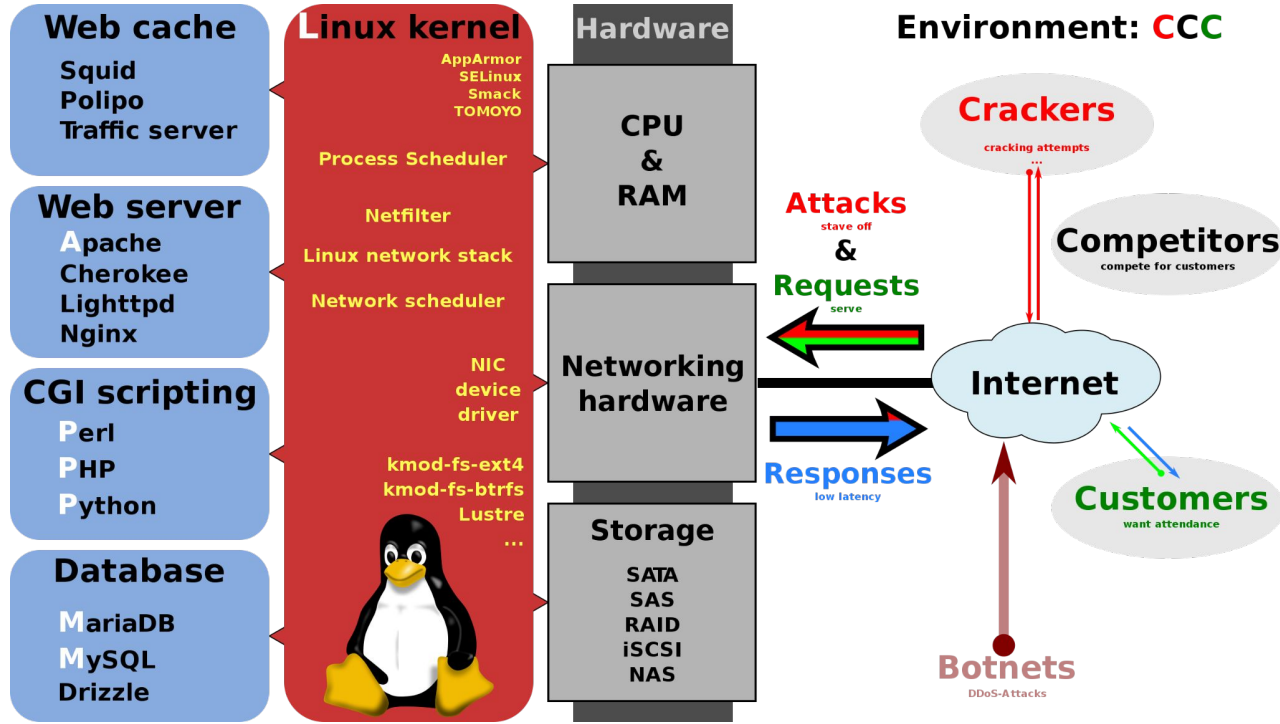
- absent → rimuove file o directory (ricorsivamente)
- directory → crea una directory e le relative directory intermedie (se non già presenti sul sistema)
- link → creazione di un link simbolico
- touch → crea un file vuoto

Ansible - Modulo file

Rimuovere un file o una directory (in modo ricorsivo).

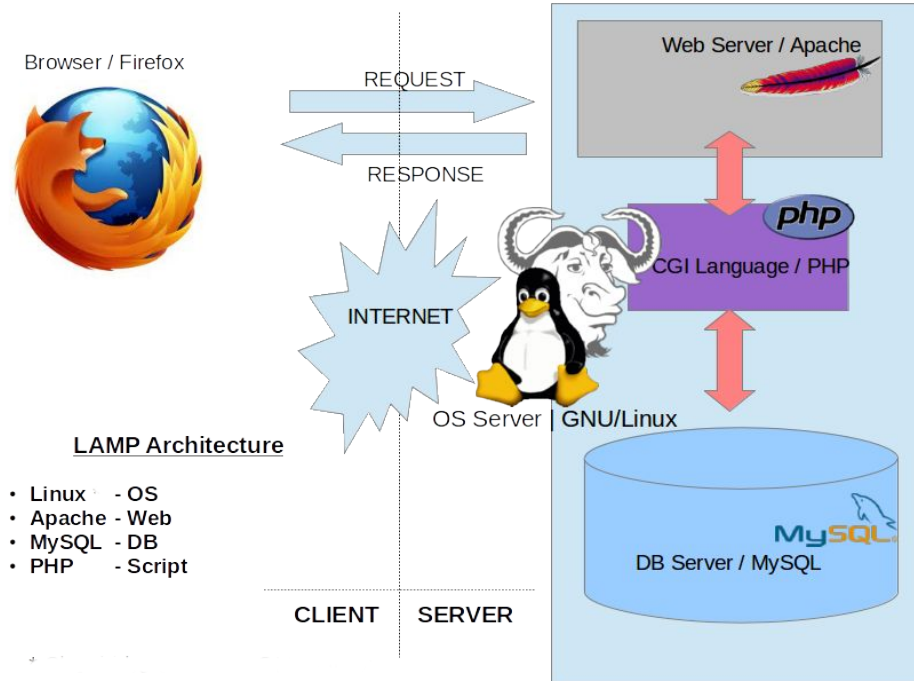
- name: Remove file (delete file)
 ansible.builtin.file:
 path: /etc/foo.txt
 state: absent
- name: Recursively remove directory
 ansible.builtin.file:
 path: /etc/foo
 state: absent

LAMP Stack - Overview



Courtesy of Wikipedia

LAMP Stack - Overview



LAMP
Linux
Apache
PHP
MySQL

Esiste anche una variante LEMP che utilizza nginx al posto di apache

Utilizzato per fornire una gran parte di applicazioni web dinamiche che utilizzano php, un chiaro esempio di three-tier application.

Courtesy of Wikipedia

LAMP stack - Ansible

Possiamo pensare di automatizzare il deploy di uno stack LEMP/LAMP utilizzando Ansible.

Soluzione utile alla configurazione di una macchina virtuale, pronta per essere utilizzata da un utente su un provider di Cloud Computing?

Di cosa abbiamo bisogno per creare una playbook?

- Installare il server apache
- Un php processor
- Un database server mysql / mariadb
- Una serie di configurazioni, che possono scalare in base alla complessità e alle richieste (utente mysql con privilegi specifici, firewall, etc...) → richiede tempo e una conoscenza approfondita di tutte le componenti

LAMP stack - Apache Conf

Come per nginx, possiamo definire un virtual host apache (example2.com) utilizzando il templating di Ansible.

```
<VirtualHost *:80>
    ServerName {{ http_host }}
    DocumentRoot /var/www/{{ http_host }}

    <Directory /var/www/{{ http_host }}>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
    </Directory>

    <FilesMatch "\.php$">
        # 2.4.10+ can proxy to unix socket
        SetHandler
"proxy:unix:/var/run/php/php8.2-fpm.sock|fcgi://localhost"
    </FilesMatch>
```

```
# Deny access to .htaccess files
    <FilesMatch "^\.ht">
        Require all denied
    </FilesMatch>
    ErrorLog ${APACHE_LOG_DIR}/{{ http_host }}_error.log
    CustomLog ${APACHE_LOG_DIR}/{{ http_host }}.com_access.log combined
</VirtualHost>
```

LAMP stack - php page

Possiamo poi creare una pagina php di esempio, o utilizzare un vostro progetto:

```
#index.php
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h1>My first PHP page</h1>
```

```
<?php
```

```
echo "Hello World!";
```

```
?>
```

```
</body>
```

```
</html>
```


LAMP stack - php page 2

In alternativa, possiamo scaricare con **git** una sito php:

<https://github.com/banago/simple-php-website>

LAMP stack - vars

Dobbiamo poi definire le variabili che saranno utilizzate nel template e nella playbook, in un file che chiameremo default.yml

```
#default.yml
```

```
---
```

```
mysql_root_password: "password_fortissima"
```

```
# un baco della sicurezza non banale
```

```
app_user: "root"
```

```
http_host: "html"
```

```
http_port: "80"
```

LAMP stack - Playbook 1

Possiamo poi definire la nostra playbook, specificando tutti i task.

```
#lamp.yml
```

```
- hosts: web
  become: true
  vars_files:
    - vars/default.yml
  tasks:
    #Apache packages -- manca upload repository
    - name: Install LAMP Packages
      apt: name={{ item }} update_cache=yes state=latest
        loop: [ 'apache2', 'mariadb-server', 'python3-pymysql', 'php',
'php-mysql', 'libapache2-mod-php' ]
```

LAMP stack - Playbook 2

```
# Apache2 Configuration
- name: Sets Apache2 conf file
  template:
    src: "files/apache2.conf.j2"
    dest:
"/etc/apache2/sites-available/{{
http_host }}.conf"

- name: Enables the new site via
symlink
  file:
    src:
"/etc/apache2/sites-available/{{
http_host }}.conf"
    dest:
"/etc/apache2/sites-enabled/{{
http_host }}.conf"
  state: link
  notify: Reload Apache2
```

```
- name: Ensure required Apache modules
are enabled
  apache2_module:
    name: "{{ item }}"
    state: present
  with_items:
    - proxy
    - proxy_fcgi
    - rewrite
  notify: Restart Apache2

- name: Delete old site directory if
present
  file:
    path: /var/www/{{ http_host }}/
    state: absent
```

LAMP stack - Playbook 3

```
# Copy the mysql credential file
to /root/.my.cnf
- name: Create mysql credential
file
  template:
    src: "files/mysql_conf_file.j2"
    dest: "/root/.my.cnf"

# MySQL set root password
- name: Sets the root password
mysql_user:
  name: root
  password: "{{ mysql_root_password }}"
}}

  login_unix_socket:
/var/run/mysqld/mysqld.sock
```

```
# Sets Up a simple PHP website
download from Github in the files dir
- name: Sets Up PHP Info Page
copy:
  src: "files/simple-php-website/"
  dest: "/var/www/{{ http_host }}"
  owner: "{{ ansible_user }}"
  group: "{{ ansible_user }}"
  mode: 0644

handlers:
  - name: Reload Apache2
    service:
      name: apache2
      state: reloaded
  - name: Restart Apache2
    service:
      name: apache2
      state: restarted
```

LAMP stack - Playbook - mysql configuration

La creazione di questo file è un hack importante!

```
# Example .my.cnf file for setting the root password new credentials
from the file.
[client]
user=root
password={{ mysql_root_password }}
```

Ansible - Basic conditional operations

Ansible può eseguire anche operazioni condizionali per eseguire task differenziate, definire gruppi di host, etc...

Sebbene l'utilizzo delle [operazioni condizionali in Ansible](#) è un'operazione dispendiosa, legata anche alla scrittura di template Jinja2, possiamo vedere alcune operazioni basilari come l'utilizzo dell'operatore **when**.

When viene utilizzato come operatore condizionale per un singolo task, ad es. Per eseguire un task solamente se l'host sta utilizzando una distribuzione Debian.

Ansible - Basic conditional operations

tasks:

- name: Configure SELinux to start mysql on any port

 - ansible.posix.seboolean:

 - name: mysql_connect_any

 - state: true

 - persistent: yes

 - when: ansible_selinux.status == "enabled"

 - # all variables can be used directly in conditionals without double curly braces

Questo task viene eseguito solamente nel caso in cui SELinux risulti abilitato.

Ansible - conditional based on variables

tasks:

```
- name: Run if "epic" or "monumental"  
is true
```

```
    ansible.builtin.shell: echo "This  
certainly is epic!"
```

```
    when: epic or monumental | bool
```

```
- name: Run if "epic" is false
```

```
    ansible.builtin.shell: echo "This  
certainly isn't epic!"
```


```
    when: not epic
```

Variabili definite in un file globale o incluso all'interno di una playbook.

vars:

```
    epic: true
```


```
    monumental: "yes"
```



Si guardi anche il modulo shell, utile per eseguire comandi su un host di tipo UNIX

Ansible - ansible_facts

Vi ricorda qualcosa?



Si possono specificare anche operazioni condizionali sulla base di `ansible_facts`. I facts sono attributi individuali associati agli host come: indirizzi IP, sistema operativo, stato di un file system etc...

Possiamo visualizzare i facts associati a un host particolare con il seguente comando:

```
$ ansible web -m setup
```

O includendo il task in una playbook

```
- name: Show facts available on the system
  ansible.builtin.debug:
    var: ansible_facts
```

Ansible - ansible_facts

Possiamo poi utilizzare i facts all'interno di un conditional per personalizzare un nostro task:

tasks:

- name: Shut down CentOS 6 systems
ansible.builtin.command: /sbin/shutdown -t now
when:
 - ansible_facts['distribution'] == "CentOS"
 - ansible_facts['distribution_major_version'] == "6"

Ansible - ansible_facts

```
---
- hosts: webservers
  remote_user: root
  vars_files:
    - "vars/common.yml"
    - [ "vars/{{ ansible_facts['os_family'] }}.yaml",
        "vars/os_defaults.yml" ]
  tasks:
    - name: Make sure apache is started
      ansible.builtin.service:
        name: '{{ apache }}'
        state: started
```

```
---
# for vars/RedHat.yml
apache: httpd
somethingelse: 42
```

Nella playbook a sinistra, il nome di un file dipende dal tipo di **os_family**, che posso reperire tramite `ansible_facts`.

Ansible - ansible_facts

Stop a service only if the service exists:

```
tasks:
```

- name: Get the list of services
service_facts:
- name: Stop service if it exists
systemd:
 name: apache2
 state: stopped
when: "'apache2.service' in services"