Playbooks

CLARUSWAY©
WAY TO REINVENT YOURSELF

Table of Contents

1

Intro to Playbooks

CLARUSWAY©
WAY TO REINVENT YOURSELF

Intro to Playbooks

- **Playbooks** are the basis for a really simple configuration management and multi-machine deployment system, unlike any that already exist, and one that is very well suited to deploying complex applications.
- **Playbooks** can declare configurations, but they can also orchestrate steps of any manual ordered process, even as different steps must bounce back and forth between sets of machines in particular orders.

```
- name: Install and Configure MySQL
  hosts: db-server
  tasks:
    - name: Install Pre-Requisites
      yum: name=pre-req-packages state=present

    - name: Install MySQL Packages
      yum: name=mysql state=present

    - name: Start MySQL Service
      service: name=mysql state=started

    - name: Configure Database
      mysql_db: name=db1 state=present
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Intro to Playbooks

The goal of a play is to map a group of hosts to some well defined roles, represented by things ansible calls tasks. At a basic level, a task is nothing more than a call to an ansible module.

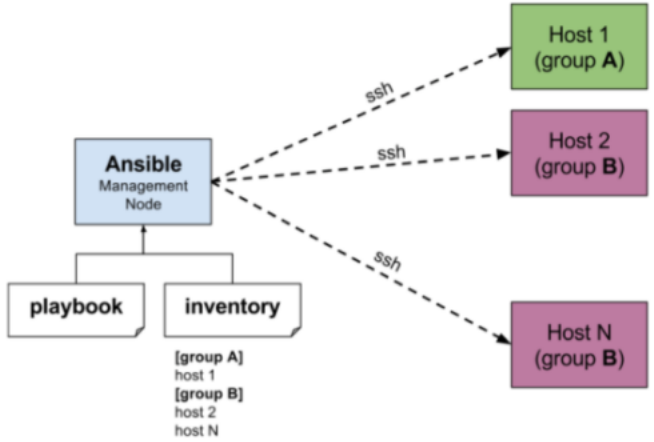
```yaml
-
  name: Play 1
  hosts: localhost
  tasks:
   - name: Execute command 'date'
     command: date

   - name: Execute script on server
     script: test_script.sh

   - name: Install httpd service
     yum:
      name: httpd
      state: present

   - name: Start web server
     service:
      name: httpd
      state: started
```

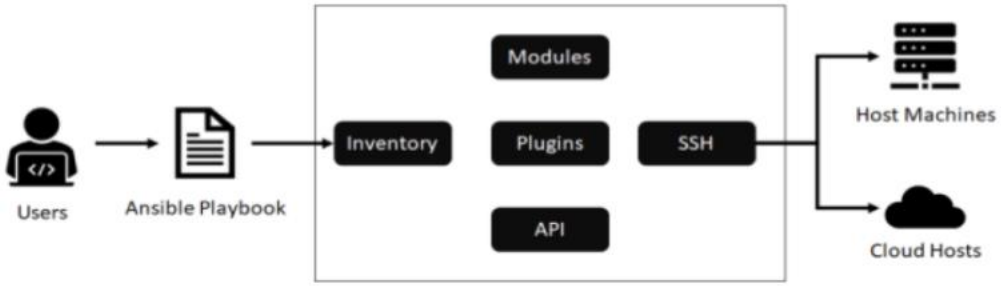
2 Hosts and Users



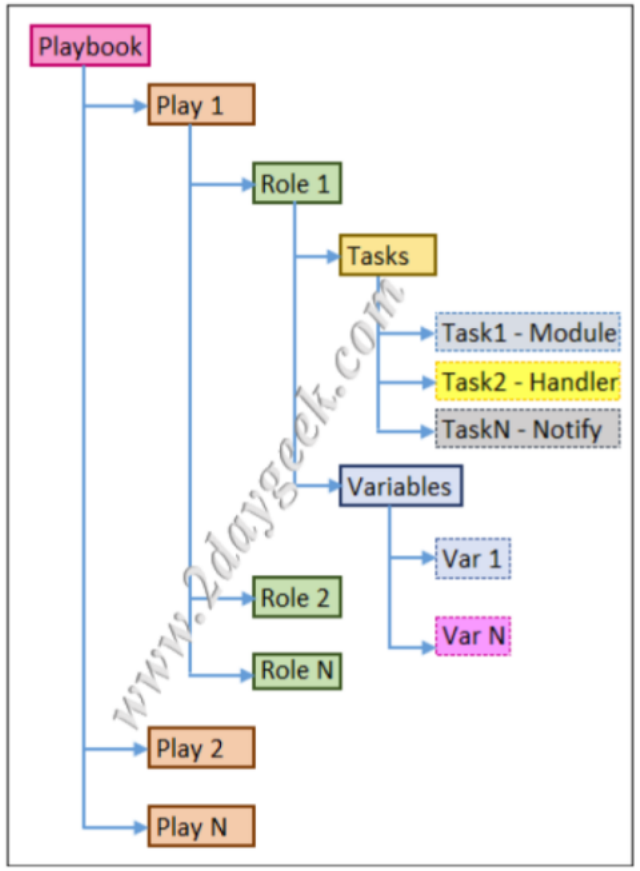
CLARUSWAY©
WAY TO REINVENT YOURSELF

Hosts and Users

- For each play in a playbook, you get to choose which machines in your infrastructure to target and what remote user to complete the steps (called tasks) as.
- The host defined in the inventory file must match the host used in the playbook and all connection information for the host is retrieved from the inventory file.



3 Tasks



CLARUSWAY©
WAY TO REINVENT YOURSELF

Tasks

- Each play contains a list of tasks. Tasks are executed in order, one at a time, against all machines matched by the host pattern, before moving on to the next task.
- The goal of each task is to execute a module, with very specific arguments. Variables can be used in arguments to modules.

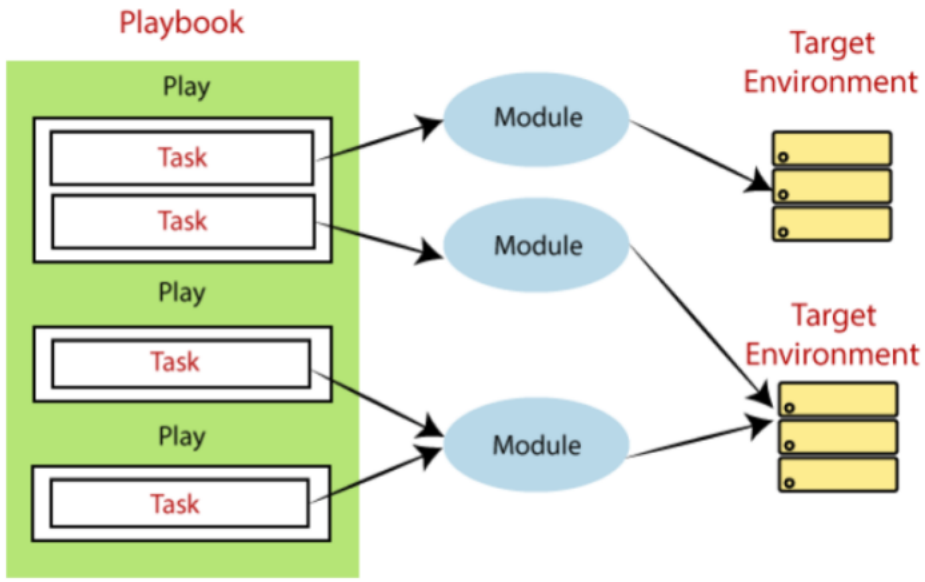
```
# Simple Ansible Playbook1.yml
-
  name: Play 1
  hosts: localhost
  tasks:
    - name: Execute comand "date"
      command: date

    - name: Execute script on server
      script: test.sh

    - name: Install httpd package
      yum:
        name: httpd
        state: present

    - name: Start web server
      service:
        name: httpd
        state: started
```

4 Modules



CLARUSWAY©
WAY TO REINVENT YOURSELF

Modules

- **Modules** (also referred to as “task plugins” or “library plugins”) are discrete units of code that can be used from the command line or in a playbook task.
- Ansible executes each module, usually on the remote target node, and collects return values.
- Modules should be **idempotent**, and should avoid making any changes if they detect that the current state matches the desired final state.

playbook.yml

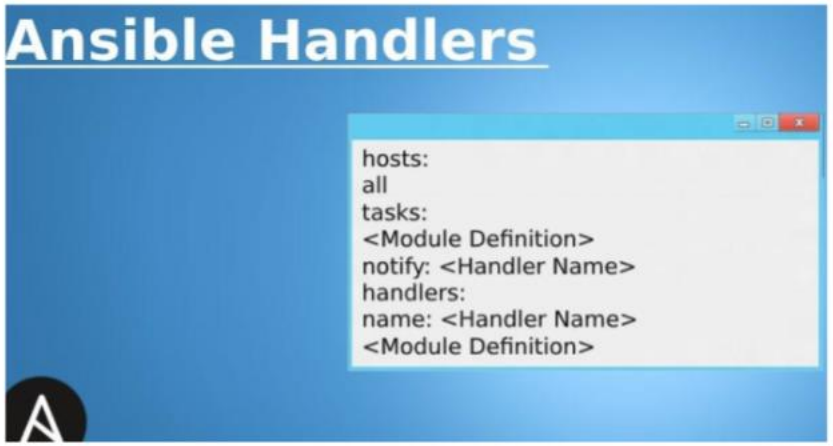
```yaml
-
  name: Play 1
  hosts: localhost
  tasks:
   - name: Execute command ‘date’
     command: date

   - name: Execute script on server
     script: test_script.sh

   - name: Install httpd service
     yum:
      name: httpd
      state: present

   - name: Start web server
     service:
      name: httpd
      state: started
```

5 Handlers

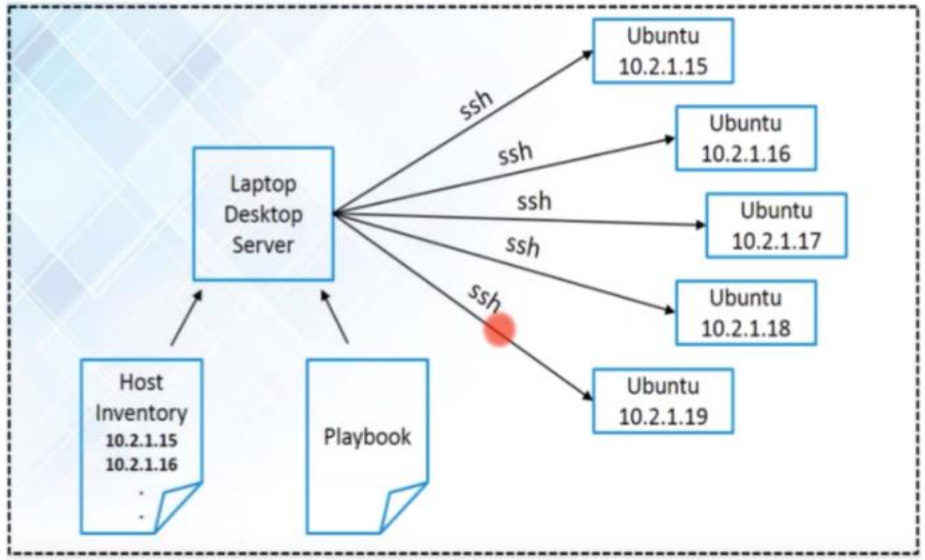


Handlers

Handlers are lists of tasks, not really any different from regular tasks, that are referenced by a globally unique name, and are notified by notifiers. If nothing notifies a handler, it will not run.

```
- hosts: webservers1
  user: root
  tasks:
  - name: test copy
    copy: src=/root/a.txt dest=/mnt
    notify: test handlers
  handlers:
  - name: test handlers
    shell: echo "abcd" >> /mnt/a.txt
~
```

6 Inventory File



Inventory File

- Ansible works against multiple managed nodes or “hosts” in your infrastructure at the same time, using a list or group of lists know as inventory.
- The default location for inventory is a file called /etc/ansible/hosts.
- You can specify a different inventory file at the command line using the -i < path > option.

Inventory Files

```
$ app.inv
[webservers]
www1.example.com
www2.example.com

[appservers]
app1.example.com
app2.example.com

[memcached]
memcached.example.com

[redis]
redis.example.com

[dbservers]
db0.example.com
```

7 Variables

Variables

- **Variables** are used to store values that varies with different items.

```
[webservers]
web1 ansible_host=3.85.110.235 ansible_user=ec2-user ansible_shh_pass=P@abcd
web2 ansible_host=3.88.62.253 ansible_user=ec2-user ansible_shh_pass=P@1234

[dbservers]
db1 ansible_host=3.85.110.235 ansible_user=ec2-user ansible_shh_pass=P@Defne
```

Playbook.yml

```
-
  name: Add DNS server to resolv.conf
  hosts: webservers
  vars:
    dns_server: 10.1.250.10

  tasks:
    -lineinfile:
      path: /etc/resolv.conf
      line:'nameserver {{ dns_server }}'
```

OR

```
#Sample variable file - web.yml

dns_server: 10.1.250.10
```

8 Conditionals

Conditionals

```yaml
- name: Install NGNIX
  hosts: webservers
  tasks:
  - name: Install NGNIX on Redhat
    yum:
      name: ngnix
      state: present
    when: ansible_os_family == "RedHat"

  - name: Install NGINIX on Debian
    apt:
      name: nginx
      state: present
    when: ansible_os_family == "Debian" and ansible_distribution_version == "16.04"
```

9 Loops

CLARUSWAY©
WAY TO REINVENT YOURSELF

Conditionals

```yaml
-
  name: 'Install required packages'
  hosts: webservers
  tasks:
    -
      yum:
        name: '{{ item }}'
        state: present
      loop:
        - httpd
        - binutils
        - glibc
        - sysstat
        - unixODBC
        - mongodb
        - nodejs
        - grunt
```