Chapter: 07 Managing large projects

selecting hosts with hosts patterns.

Host patterns are used to specify the hosts to target by a play or ad-hoc command.

→ By mentioning single manged host listed in the inven. If IP is mentioned in inventory; IP can be used to call host in playbook.
If hostname —"— ; hostname —"—
DNS is not resolved automatically of M.N.

→ By mentioning the group from the inventory, it will act on all M.N inside those group.
all ⇒ special group ⇒ all managed node from inventory.
Ungrouped ⇒ special group ⇒ all M.N that are individual & not members of any group.

→ By Wildcards.
'*' ⇒ all

As wildcards can be used to call out hosts; it is possible that there may be special characters in hostnames:
To overcome any failure; it is good & prefered that hosts must be mentioned in single quotes. so it may parse correctly.

```
---
- hosts: 'development'
```

→ By '- hosts: '*.example.com' to act as wildcard char and include all hosts ending with example.com

- hosts: 'datacenter*'
includes all hosts starting with datacenter.

→ By list of hosts seperated by comma
- hosts: node1, node2, node3
- hosts: group1, group2, group3

→ By Mixture of host groups, wildcard & hosts
- hosts: group1, node1, data*

→ By using logical AND to find the hosts which are member of both group. (only common hosts)
- hosts: group1, &group2
group1, &group2 ⇒ group2, &group1

→ By using logical NOT to exclude specific host or group.
- hosts: all, !node1 ⇒ all except node1
all, !node1 ⇒ !node1, all

Managing Dynamic Inventories.

external directory service = Zabbix, FreeIPA, Active Directory
Installation servers = cobbler, Red Hat Satellite
Cloud service = AWS EC2, Openstack deployment, V.M infra.

- Ansible supports dynamic inventory scripts. These scripts are executable programs that collect information from some external source & output the inventory in JSON format.

To write a dynamic inventory; start the script with interpreter line (#! /usr/bin/python : example)

- Ansible supports multiple inventory.
Inventory files should not depend on the other inventory files or scripts in order to resolve
'-i' is used to mention inventory
- Ansible supports the use of multiple inventories in the same run. If the location of the inventory is a directory, then all the inventory files in the directory are combined to determine inventory.

- Ansible ignores files in an inventory directory if they end with certain suffixes. This can be controlled with the inventory_ignore_extensions. directive in the Ansible configuration file.

Configuring Parallelism.

Normally, all hosts must successfully complete a task before any host starts the next task in the play.
The maximum number of simultaneous connections that ansible makes is controlled by the forks parameter in the ansible configuration file.
it is set to '5' by default.

$ grep fork ansible.cfg.
$ grep $ansible-config dump | grep -i forks
$ ansible-config list | grep -i forks.

} To verify fork value.

Default forks ⇒ 5 forks ⇒ Ansible will run 1st task on first 5 M.N & then on other 5 M.N & so on until all M.N are completed. After that Ansible will run 2nd task, and so on until play is end.

If C.N is managing all linux M.N then fork value can be set to 100.

Default fork can be overridden on CLI by '-f', '--forks' options / or on ansible.cfg forks = value.

Serial

To overcome sudden updating / restarting of all servers, M.N at once, 'serial' keyword can be used, this will ensure, Each batch of hosts will be run through the entire play before the next batch is started.

```
- name: Runing serial play
  hosts: all
  serial: 2
  tasks:
```

Runs entire play on first 2 M.N then proceed with next 2 & so on.

If number is not mentioned then be default it is '1'

If something is wrong while running first 2 M.N then the entire play fails & is aborted.

Serial keyword can also be specified in percentage.

Including & Importing files.

To bring content into a playbook, you can include content or you can import content.

include = dynamic operation
import = static operation

import_playbook ⇒ Master playbook that imports one or more additional playbooks.

→ Content being imported is a complete playbook.
→ Can not be used inside a play
→ Used at top level of a playbook

```
- name: importing play1.yml
  import_playbook: play1.yml

- name: importing play2.yml
  import_playbook: play2.yml
```

→ You can interleave plays in your master playbook with imported playbooks.

importing & including tasks

You can import or include a list of tasks from a task file into a play.

```
# cat task.yml
- yum:
    name: httpd
    state: present
```

```
---
- name: Importing task file
  hosts: all
  tasks:
    - import_tasks: task.yml
```

task file can be imported into a play inside a playbook.

① Conditional statement (when, if) are applied on each task in the imported_tasks, if used.
② loops can't be used with import_tasks feature.
③ If variable is used to call out import_tasks file then you can not use host or group inventory variables.

Include task file.

```
---
- name: Including task file.
  hosts: all
  tasks:
    - include_tasks: task1.yml
```

The include_tasks feature does not process content in the playbook until the play is running and that part of the play is reached.

→ Does not show tasks in:
ansible-playbook --list-tasks.

→ Cannot use:
ansible-playbook --start-at-tasks

→ Cannot use notify statement to trigger a handler name that is in an included task file.

To manage structure & components on ansible, it is better to make a directory of tasks & include all tasks in them; and call out them while importing or including.

Defining variables.

```
---
- name: Install the {{ package }} package
  yum:
    name: "{{ package }}"
    state: latest
- name: start the {{ service }} service
  service:
    name: "{{ service }}"
    enabled: true
    state: started.
```

$ vim playbook.yml

```
  :
tasks:
  - name: Import task & variables.
    import_tasks: task.yml
      vars:
        package: httpd
        service: httpd.
```