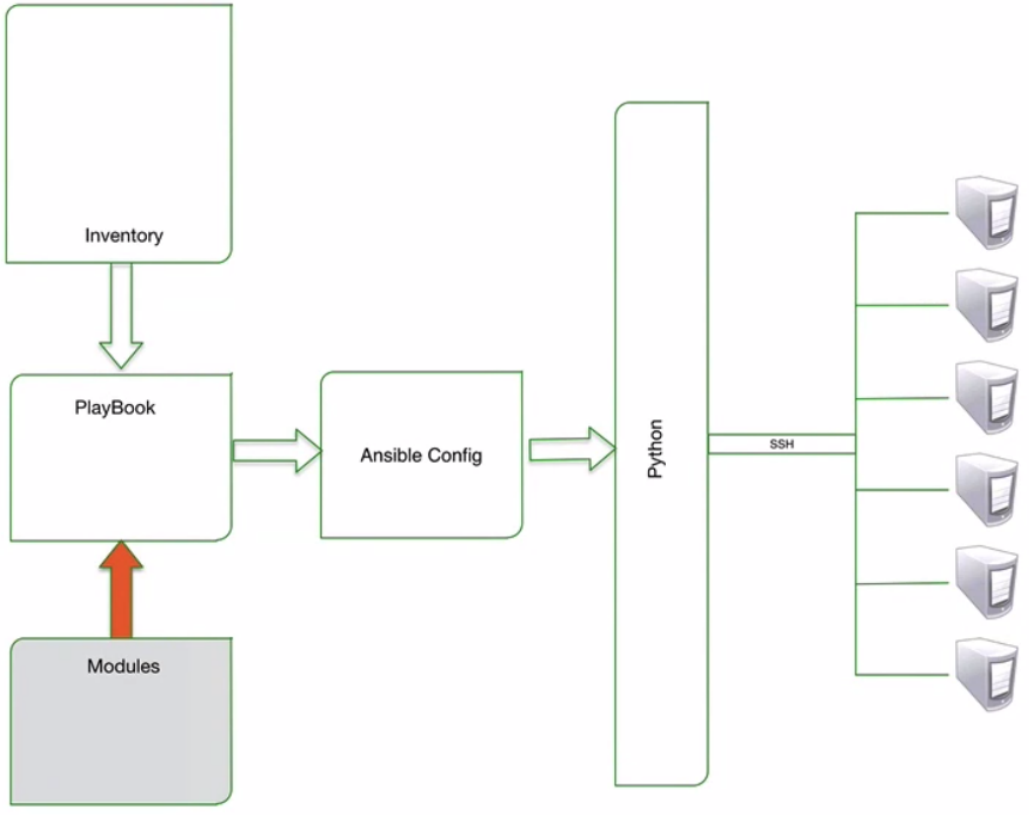

# Ansible Modules, Plays and Playbook

**By –** **Yathish Nagaraj**
Infra Dev Specialist
Automation PS Team

# Ansible Modules



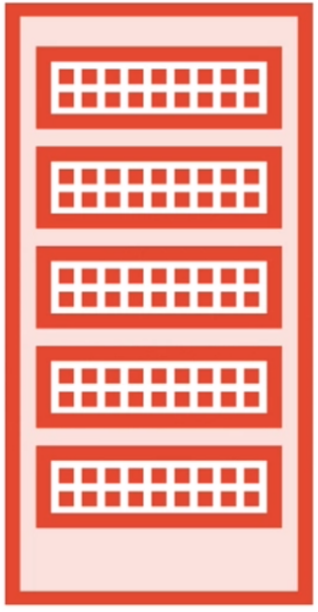
# Modules Docs

```
$ ansible-doc -l
```

```
$ ansible-doc -s
<name>
```

```
$ ansible-doc <name>
```

# Modules Categories



- Manage Servers
- Deploy Configurations

# Modules Categories

- Configure network equipment

# Modules Categories


- Maintain virtual servers

# Modules Categories

- Manage databases and tables

# Modules Categories

- Deploy load-balancer configurations

# Copy Modules

- Copies a file from local box to remote system
- Has “backup” capability
- Can do validation remotely

# Fetch Modules

- Pulls a file from remote host to local system
- Can use md5 checksums to validate

# Apt Modules

- Manages installed applications on Debian-based systems
- Can install, update, or delete packages
- Can update entire system

# Yum Modules

- Manages installed applications on Redhat-based systems
- Can install, update, or delete packages
- Can update entire system

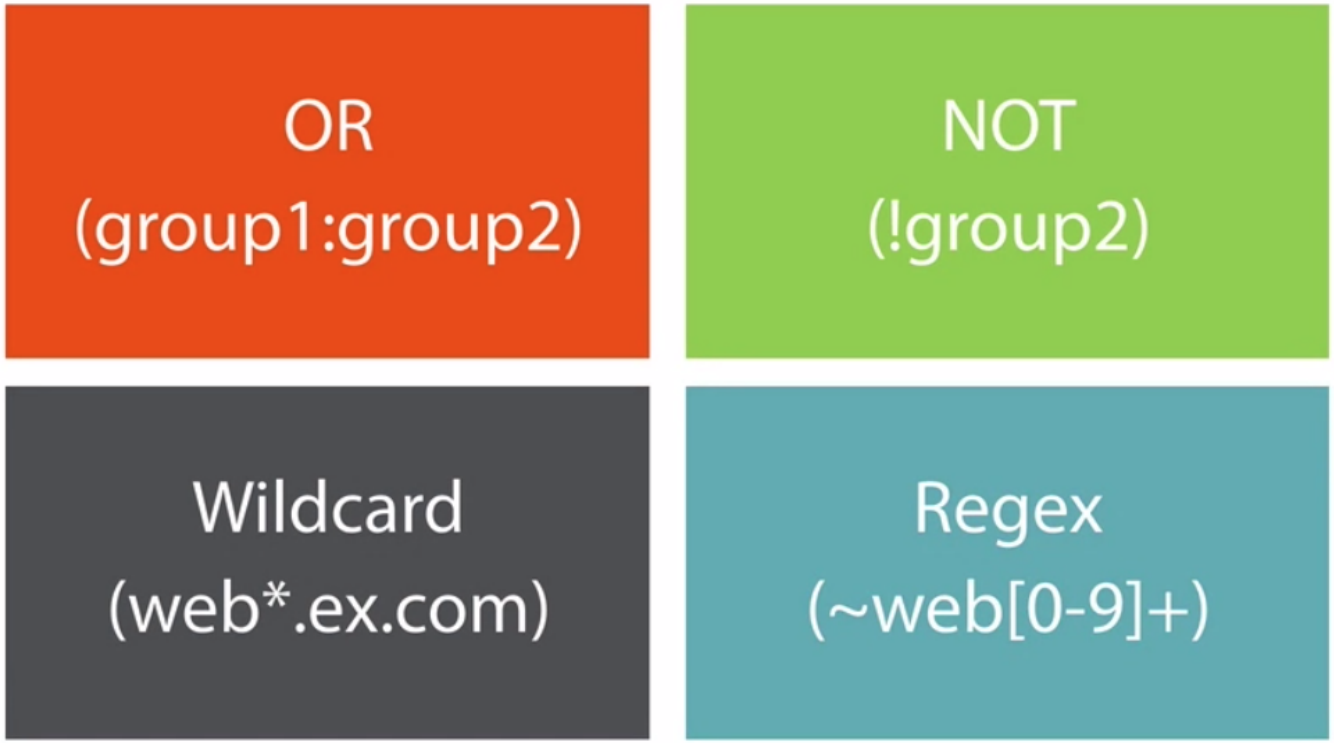
# Service Modules

- Can stop, start, or restart services
- Can enable services to start on boot

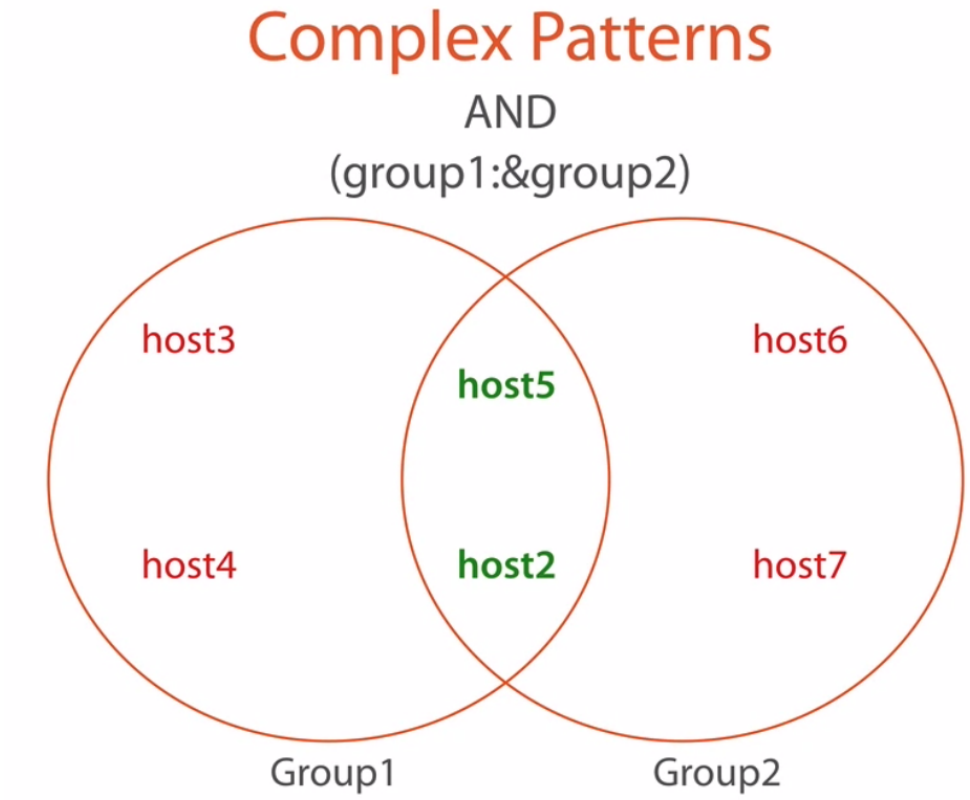
# Demonstration: Using Modules to Install/Start

- Browse module documentation
- Install Web Server (Yum module)
- Start Web Server (Service module)
- Install DB Server (Yum module)
- Start DB Server (Service module)
- Stop Firewalls (Service module)

# Host/Group Target Patterns



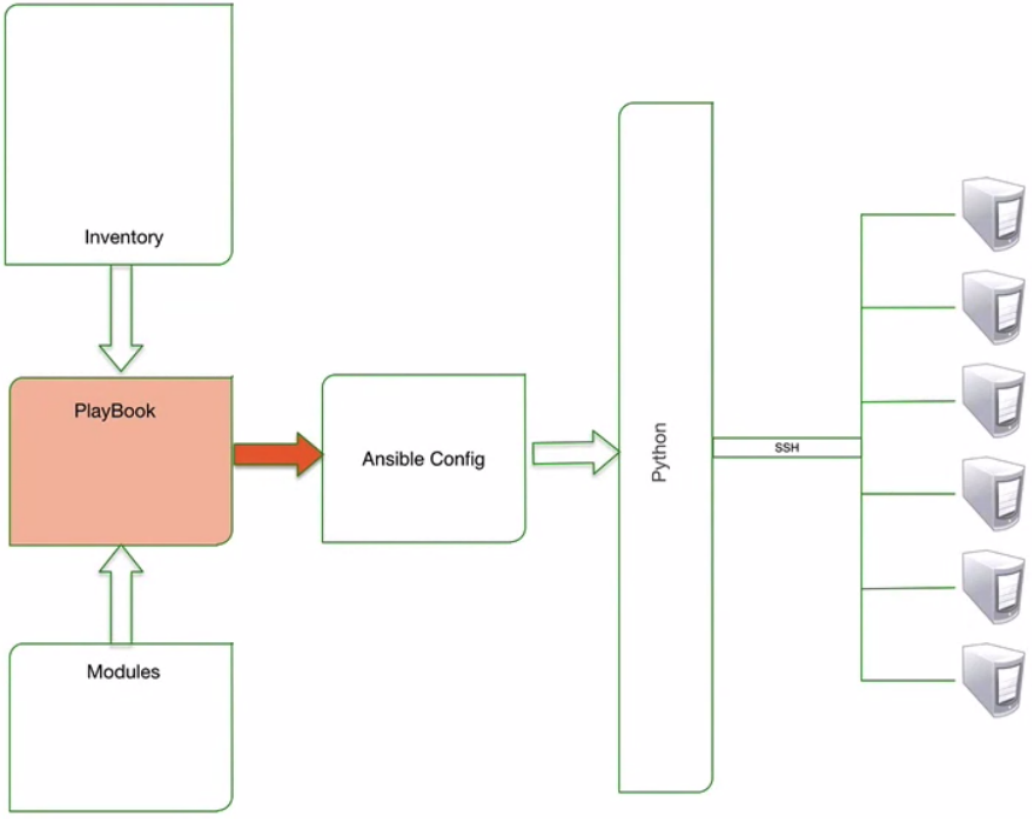
# Host/Group Target Patterns



# Demonstration: Using Setup Module

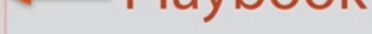
- Gather facts on remote systems
- Used in Playbooks

# Ansible Plays and Playbooks



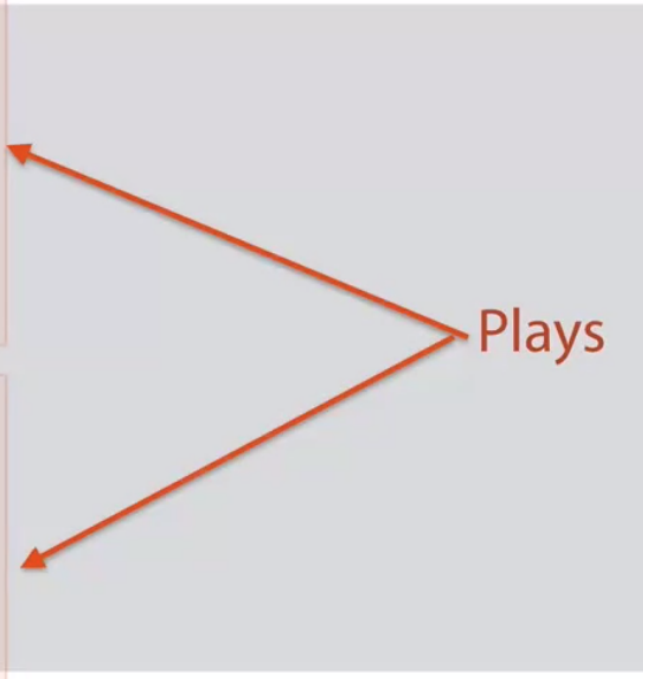
# Ansible Plays and Playbooks

```
---
- hosts: webservers
  remote_user: root
  tasks:
  - name: Install Apache
    yum: name=httpd state=present
  - name: Start Apache
    service: name=httpd state=started

- hosts: dbservers
  remote_user: root
  tasks:
  - name: Install MySQL
    yum: name=mysql-server state=present
  - name: Start MySQL
    service: name=mysqld state=started
```



# Ansible Plays and Playbooks

```
---
- hosts: webservers
  remote_user: root
  tasks:
  - name: Install Apache
    yum: name=httpd state=present
  - name: Start Apache
    service: name=httpd state=started

- hosts: dbservers
  remote_user: root
  tasks:
  - name: Install MySQL
    yum: name=mysql-server state=present
  - name: Start MySQL
    service: name=mysqld state=started
```



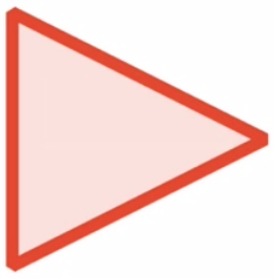
# Ansible Plays and Playbooks

```
---
- hosts: webservers
  remote_user: root
  tasks:
  - name: Install Apache
    yum: name=httpd state=present
  - name: Start Apache
    service: name=httpd state=started
```

Global Play Declaration

```
- hosts: webservers
  vars:
    git_repo: https://github.com/repo.git
    http_port: 8080
    db_name: wordpress
  sudo: yes
  sudo_user: wordpress_user
  gather_facts: no
```

# Ansible Plays and Playbooks



Execution of playbooks:

```
$ ansible-playbook playbook.yml
```

# Ansible Plays and Playbooks

## Retrying Failed Host Executions

```
PLAY RECAP *********************************************************************
        to retry, use: --limit @/home/vagrant/ping.retry

db1                        : ok=0    changed=0    unreachable=1    failed=0
web1                       : ok=2    changed=0    unreachable=0    failed=0
```

# Demonstration: Our first Playbook

Write a playbook

Add play to install web server

Add play to install db server

Add play to start services

Fail a play

Retry a failed play

# Few Advanced features of Playbooks

## Including Files

### Include Files to Extend Playbook

```
tasks:
  - include: wordpress.yml
    vars:
      sitename: My Awesome Site
  - include: loadbalancer.yml
  - include_vars: variables.yml
```

- Breaks up long playbooks
- Use to add external variable files
- Reuse other playbooks

# Few Advanced features of Playbooks

## Register Task Output

Grab output of task for another task

```
tasks:
  - shell: /usr/bin/whoami
    register: username
  - file: path=/home/myfile.txt
          owner={{ username }}
```

- Useful to use tasks to feed data into other tasks
- Useful to create custom error trapping

# Few Advanced features of Playbooks

## Debug Module

### Add debug to tasks

```
tasks:
  - debug: msg="This host is
{{ inventory_hostname }} during
execution"

  - shell: /usr/bin/whoami
    register: username
  - debug: var=username
```

- Useful to send output to screen during execution
- Helps find problems

# Few Advanced features of Playbooks

## Prompting for Input

### Prompt user during execution

```
- hosts: web1

  vars_prompt:
   - name: "sitename"
     prompt: "What is new site name?"

  tasks:
   - debug: msg="The name is {{ sitename }}"
```

- Creates Dynamic Playbooks

# Few Advanced features of Playbooks

## Playbook Handlers

- Tasks with asynchronous execution
- Only runs tasks when notified
- Tasks only notify when state=changed
- Does not run until all playbook tasks have executed
- Most common for restarting services to load changes (if changes are made)

# Few Advanced features of Playbooks

## Handlers

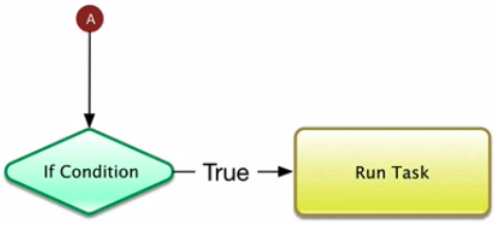
### Notify handlers from your tasks

```
tasks:
   - copy: src=files/httpd.conf
           dest=/etc/httpd/conf/
     notify:
        - Apache Restart
 handlers:
   - name: Apache Restart
     service: name=httpd state=restarted
```

- Copies config file to host
- If state=change on “COPY”, tell “Apache Restart”
- Run “Service” module.

# Few Advanced features of Playbooks

## Conditional Execution




Use the clause “when” to choose if task should run.

# Few Advanced features of Playbooks

## Conditional Clause

Choose when to execute tasks

```
tasks:
   - yum: name=httpd state=present
     when: ansible_os_family == “RedHat”

   - apt: name=apache2 state=present
     when: ansible_os_family == “Debian”
```

- Uses YUM if OS is RedHat
- Uses APT if OS is Debian

# Few Advanced features of Playbooks

## Conditional Clause Based on Output

Choose when to execute tasks

```
tasks:
  - command: ls /path/doesnt/exist
    register: result
    ignore_errors: yes

  - debug: msg="Failure!"
    when: result|failed
```

- Track whether previous task ran
- Searches JSON result for status
- Status Options:
  - success
  - failed
  - skipped

# Few Advanced features of Playbooks

## Templates

Uses Jinja2 Engine

Insert variables into static files

Creates and copies dynamic files

Deploy custom configurations

# Few Advanced features of Playbooks

## Template Module

### Modify Template and Copy

```
tasks:
  - template:
      src=templates/httpd.j2
      dest=/etc/httpd/conf/httpd.conf
      owner=httpd
```

- Takes a file with pre-defined variable names
- Inserts variable values in file
- Copies file to destination

# Few Advanced features of Playbooks

httpd.j2

```
.....

<VirtualHost *:80>
  ServerAdmin {{ server_admin }}
  DocumentRoot {{ site_root }}
  ServerName {{ inventory_hostname }}
</VirtualHost>

.....
```

# Demonstration: Advanced Playbook Usage

Add install decisions based on OS

Create template for Apache Config

Deploy configuration

Restart service if needed