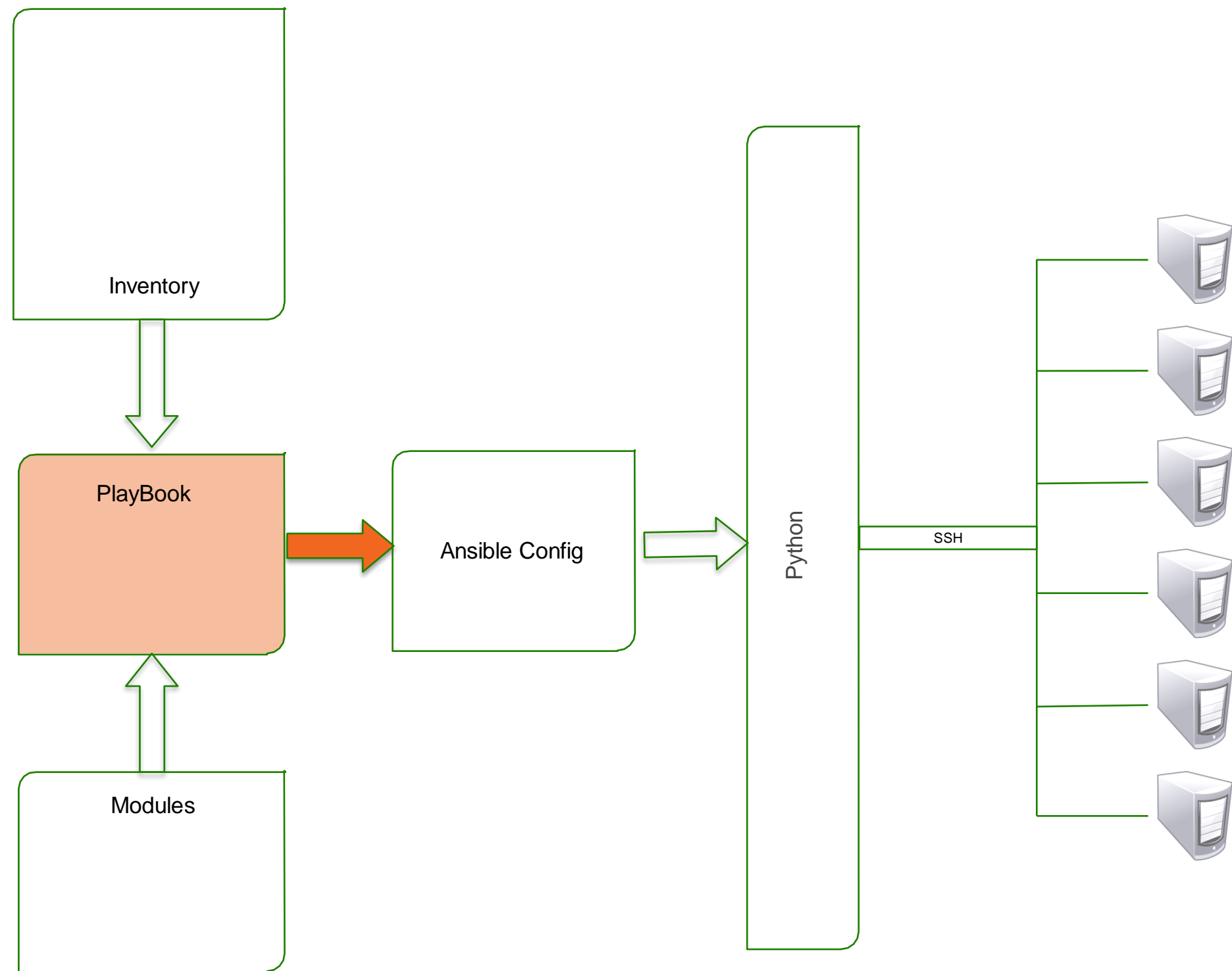
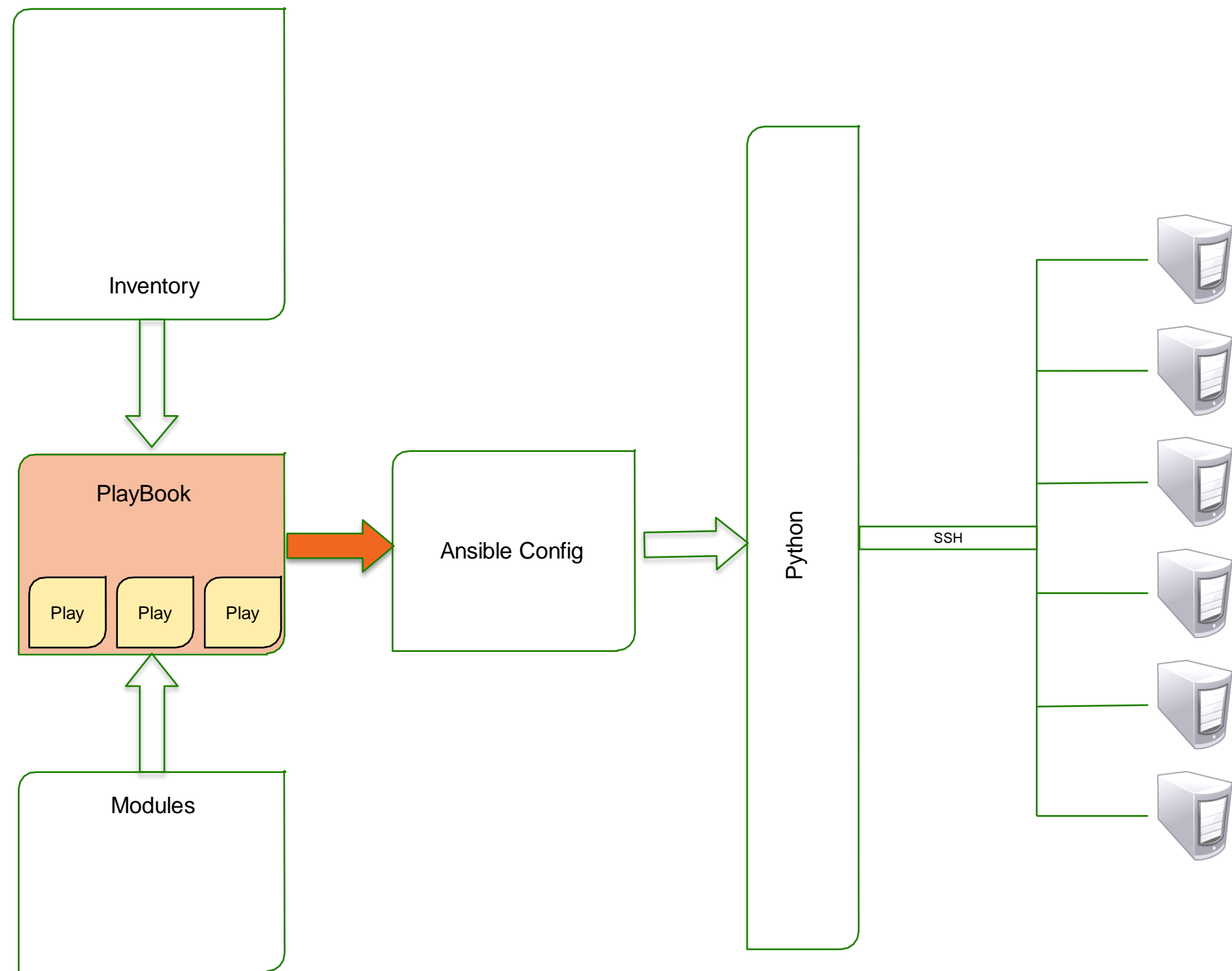
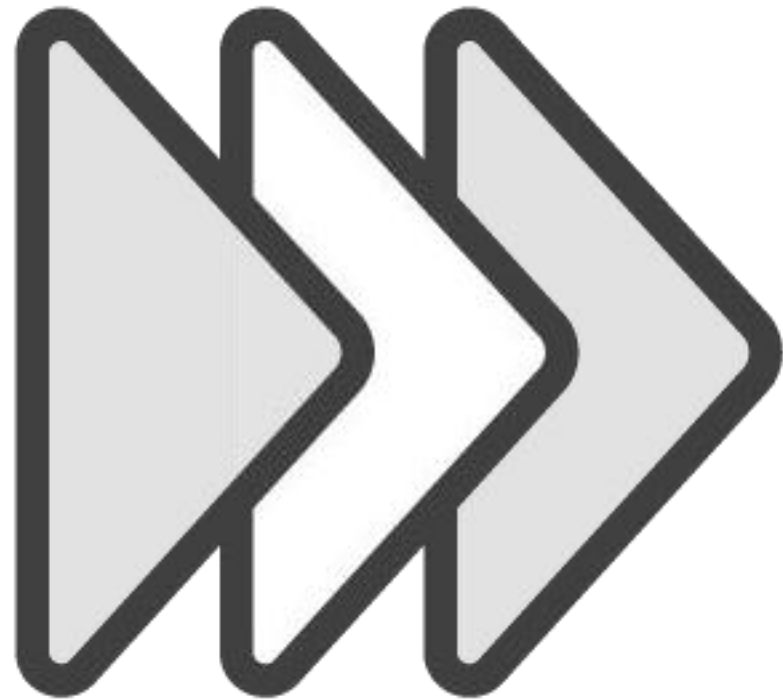


# Plays and Playbooks









Plays map hosts to tasks

A play can have multiple tasks

A playbook can have multiple plays

# Playbook Breakdown

```
---  
- 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
```



Plays

# YAML Whitespace



```
---
- 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=mysqlld state=started
```

The diagram illustrates the correct use of whitespace in YAML. Orange arrows point to the following elements:

- The first arrow points to the first dash of the first list item.
- The second arrow points to the `hosts: webservers` line.
- The third arrow points to the `remote_user: root` line.
- The fourth arrow points to the `yum: name=httpd state=present` line.
- The fifth arrow points to the `remote_user: root` line of the second list item.
- The sixth arrow points to the `yum: name=mysql-server state=present` line.

Whitespace is crucial!

# Play Breakdown

---

- hosts: webserver  
remote\_user: root  
tasks:
  - name: Install Apache  
yum: name=httpd state=present
  - name: Start Apache  
service: name=httpd state=started



# Play Breakdown

---

```
- hosts: webserver  
  remote_user: root  
  tasks:
```

Global Play Declaration

```
- name: Install Apache  
  yum: name=httpd state=present  
- name: Start Apache  
  service: name=httpd state=started
```

# Play Declarations

-

```
vars:
```

```
  git_repo: https://github.com/repo.git
```

```
  http_port: 8080
```

```
  db_name: wordpress
```

```
sudo: yes
```

```
sudo_user: wordpress_user
```

```
gather_facts: no
```

# Play Declarations

```
- hosts: webservers
```

```
sudo: yes
```

```
sudo_user: wordpress_user
```

```
gather_facts: no
```

Declare Variables per Play

# Play Declarations

```
- hosts: webservers
  vars:
    git_repo:
      https://github.com/repo.git
      http://name:password@port:port
    sudo: yes
    sudo_user: wordpress_user
  gather_facts: no
```

Declare user to run tasks

# Play Declarations

```
- hosts: webservers
  vars:
    git_repo:
      https://github.com/repo.git
      http://name:password@hostname:8080
    sudo: yes
    sudo_user:
      1
    gather_facts: no
```

Don't gather facts on  
hosts



Tasks are executed in order - top down

Tasks use modules

# Tasks

tasks:

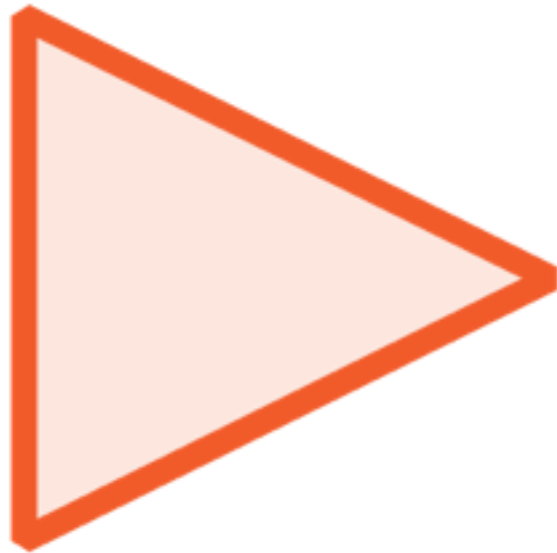
- name: Name this task for readability

module: parameters=go\_here

- name: Deploy Apache Configuration File

copy: src=/ansible/files/conf/httpd.conf

    f dest=/etc/httpd/conf/



## Execution of playbooks:

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



If a host fails a task, that host is removed from the rest of the playbook execution



# 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

# Demo: 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



# 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

All **include\*** statements are processed as they encountered during the execution of the playbook.

# Importing Files

It is possible to include playbooks inside a master playbook. For example:

```
---  
  
import_playbook: webservers.yml  
import_playbook: databases.yml
```

You can also pass variables into imports and includes:

```
tasks:  
- import_tasks: wordpress.yml wp_user=timmy  
- import_tasks: wordpress.yml wp_user=alice  
- import_tasks: wordpress.yml wp_user=bob
```

You can then use `import_tasks` or `include_tasks` to include this file in your main task list:

```
tasks:  
- import_tasks: common_tasks.yml  
# or  
- include_tasks: common_tasks.yml
```

All `import*` statements are pre-processed at the time playbooks are parsed.

# 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

# 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

# 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



# 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)



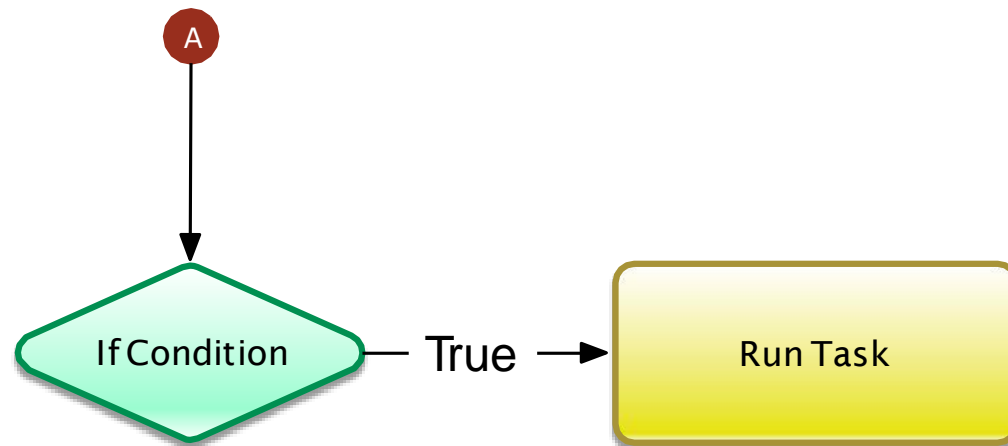
# Handlers

## 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.

# Conditional Execution



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

# 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

# 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

# Templates



Uses Jinja2 Engine

Insert variables into static files

Creates and copies dynamic files

Deploy custom configurations

# 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

# httpd.j2

... ..

```
<VirtualHost *:80>
```

```
    ServerAdmin {{ server_admin }}
```

```
    DocumentRoot {{ site_root }}
```

```
    ServerName {{ inventory_hostname }}
```

```
</VirtualHost>
```

... ..



# Demo: Playbook Controls

Add install decisions based on OS

Create template for Apache Config

Deploy configuration

Restart service if needed

