Links:

Plural sight: <https://app.pluralsight.com/library/courses/hands-on-ansible/table-of-contents>

Official Ansible Page: <https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html>

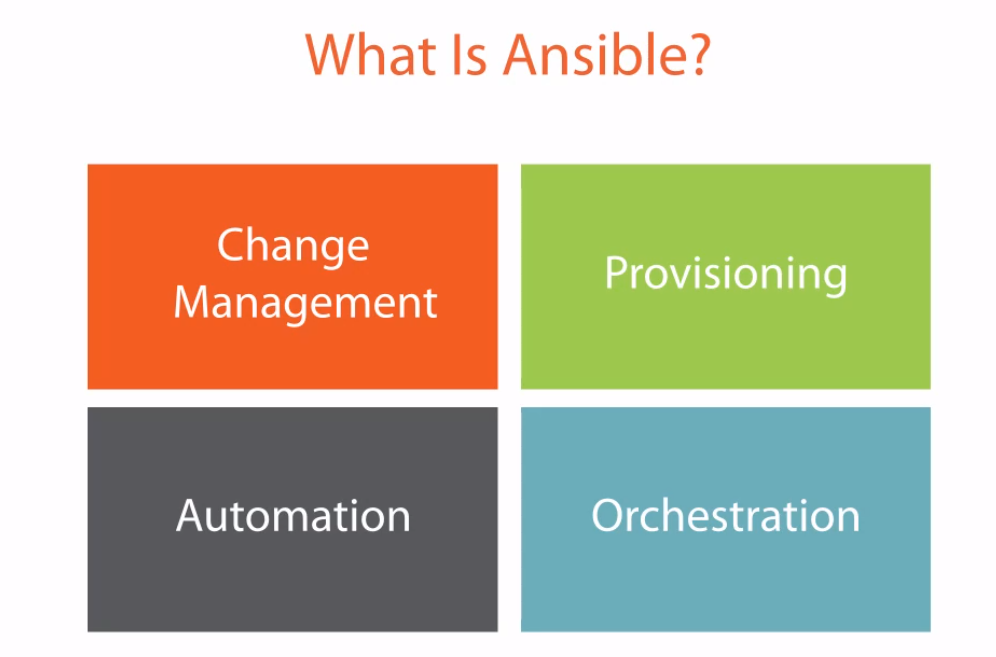
What is ansible:

Ansible is an agentless IT engine that automates different tasks:

1. Orchestration: which means, integration of multiple applications and executing them in specific order.
2. Configuration management: Consistency of all systems in the infrastructure is maintained.
3. Deployment: means deploying application on different servers.

Agentless: Unlike other configuration management tool (puppet & chef), ansible do not use any agents to complete its task. Whereas, other tool uses their agents (puppet client or chef agent) to accomplish their task.

Ansible main functions:



# Ansible concepts

These concepts are common to all uses of Ansible. You need to understand them to use Ansible for any kind of automation. This basic introduction provides the background you need to follow the rest of the User Guide.

* [Control node](https://docs.ansible.com/ansible/latest/user_guide/basic_concepts.html#control-node)
* [Managed nodes](https://docs.ansible.com/ansible/latest/user_guide/basic_concepts.html#managed-nodes)
* [Inventory](https://docs.ansible.com/ansible/latest/user_guide/basic_concepts.html#inventory)
* [Modules](https://docs.ansible.com/ansible/latest/user_guide/basic_concepts.html#modules)
* [Tasks](https://docs.ansible.com/ansible/latest/user_guide/basic_concepts.html#tasks)
* [Playbooks](https://docs.ansible.com/ansible/latest/user_guide/basic_concepts.html#playbooks)

## [Control node](https://docs.ansible.com/ansible/latest/user_guide/basic_concepts.html#id1)

Any machine with Ansible installed. You can run commands and playbooks, invoking /usr/bin/ansible or /usr/bin/ansible-playbook, from any control node. You can use any computer that has Python installed on it as a control node - laptops, shared desktops, and servers can all run Ansible. However, you cannot use a Windows machine as a control node. You can have multiple control nodes.

## [Managed nodes](https://docs.ansible.com/ansible/latest/user_guide/basic_concepts.html#id2)

The network devices (and/or servers) you manage with Ansible. Managed nodes are also sometimes called “hosts”. Ansible is not installed on managed nodes.

## [Inventory](https://docs.ansible.com/ansible/latest/user_guide/basic_concepts.html#id3)

A list of managed nodes. An inventory file is also sometimes called a “hostfile”. Your inventory can specify information like IP address for each managed node. An inventory can also organize managed nodes, creating and nesting groups for easier scaling. To learn more about inventory, see [the Working with Inventory](https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html#intro-inventory) section.

## [Modules](https://docs.ansible.com/ansible/latest/user_guide/basic_concepts.html#id4)

The units of code Ansible executes. Each module has a particular use, from administering users on a specific type of database to managing VLAN interfaces on a specific type of network device. You can invoke a single module with a task, or invoke several different modules in a playbook. For an idea of how many modules Ansible includes, take a look at the [list of all modules](https://docs.ansible.com/ansible/latest/modules/modules_by_category.html#modules-by-category).

## [Tasks](https://docs.ansible.com/ansible/latest/user_guide/basic_concepts.html#id5)

The units of action in Ansible. You can execute a single task once with an ad-hoc command.

## [Playbooks](https://docs.ansible.com/ansible/latest/user_guide/basic_concepts.html#id6)

Ordered lists of tasks, saved so you can run those tasks in that order repeatedly. Playbooks can include variables as well as tasks. Playbooks are written in YAML and are easy to read, write, share and understand. To learn more about playbooks, see [About Playbooks](https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html#about-playbooks).

-------------\*\* content of local desktop to be added here \*\* -----------------

Inventory file basic features:



**Basic test of inventory file:**

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.

The inventory file can be in one of many formats, depending on the inventory plugins you have. The most common formats are INI and YAML. A basic INI etc/ansible/hosts might look like this:

mail.example.com

[webservers]

foo.example.com

bar.example.com

[dbservers]

one.example.com

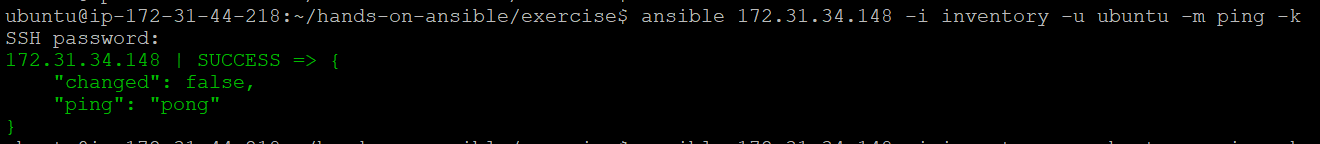
two.example.com

three.example.com

The headings in brackets are group names, which are used in classifying hosts and deciding what host you are controlling at what times and for what purpose.

After verifying that the servers are accessible to each other using ping or any other method we can start setting up our ansible setup. As a basic test we can follow below steps:

1. Create an inventory file and add IP of any remote server into it.
2. Then using ansible ping module to check its successful execution, example:



NOTE: To see what is happening in the background when ansible runs, use the debug/verbose mode. To do it, add -vvvv in the end of ansible command, for example:

**ansible 172.31.34.148 -i inventory -u ubuntu -m ping -k -vvv**

number of V means the depth of debug mode.

**Ansible Config File:**

Default location for config file depending on the installing method:

If installing Ansible from a package manager, the latest ansible.cfg file should be present in /etc/ansible, possibly as a .rpmnew file (or other) as appropriate in the case of updates. If you installed Ansible from pip or from source, you may want to create this file in order to override default settings in Ansible.

Order of search for ansible.cfg file:

* ANSIBLE\_CONFIG (environment variable if set)
* ansible.cfg (in the current directory)
* ~/.ansible.cfg (in the home directory)
* /etc/ansible/ansible.cfg

Ansible will process the above list and use the first file found, all others are ignored.

NOTE: To avoid security risks, the cfg file shouldn’t be kept in a open-to-world directory. If there is a need to do so, the authorization of the directory and file should be set properly.

CFG file parameter example:

INTERPRETER\_PYTHON: default interpreter to use for ansible. Default is /usr/bin/python.

HOST\_KEY\_CHECKING: How to handle unknown SSH keys and to enable/disable the check of fingerprints while connecting to any remote server. Default is True

FORKS: Total number of parallel operations Ansible executes. Default is 5

LOG\_PATH: path to log files. Default is no.

## [**Precedence categories**](https://docs.ansible.com/ansible/latest/reference_appendices/general_precedence.html#id1)

Ansible offers four sources for controlling its behavior. In order of precedence from lowest (most easily overridden) to highest (overrides all others), the categories are:

* Configuration settings -> defined in ansible.cfg file
* Command-line options -> specified when calling ansible from command line
* Playbook keywords -> defined inside the playbook following the playbook precedence
* Variables -> a particular variable with latest value according to precedence.

Each category overrides any information from all lower-precedence categories. For example, a playbook keyword will override any configuration setting.

Within each precedence category, specific rules apply. However, generally speaking, ‘last defined’ wins and overrides any previous definitions.

### [Playbook keywords](https://docs.ansible.com/ansible/latest/reference_appendices/general_precedence.html#id4) precedence:

Any [playbook keyword](https://docs.ansible.com/ansible/latest/reference_appendices/playbooks_keywords.html#playbook-keywords) will override any command-line option and any configuration setting.

Within playbook keywords, precedence flows with the playbook itself; the more specific wins against the more general:

* play (most general)
* blocks/includes/imports/roles (optional and can contain tasks and each other)
* tasks (most specific)

A simple example:

* **-** hosts**:** all
* connection**:** ssh
* tasks**:**
* **-** name**:** This task uses ssh.
* ping**:**
* **-** name**:** This task uses paramiko.
* connection**:** paramiko
* ping**:**

In this example, the connection keyword is set to ssh at the play level. The first task inherits that value, and connects using ssh. The second task inherits that value, overrides it, and connects using paramiko. The same logic applies to blocks and roles as well.

Ansible 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.

You can execute modules from the command line:

ansible webservers -m service -a "name=httpd state=started"

ansible webservers -m ping

ansible webservers -m command -a "/sbin/reboot -t now"

Each module supports taking arguments. Nearly all modules take key=value arguments, space delimited. Some modules take no arguments, and the command/shell modules simply take the string of the command you want to run.

From playbooks, Ansible modules are executed in a very similar way:

**-** name**:** reboot the servers

action**:** command /sbin/reboot -t now

Which can be abbreviated to:

**-** name**:** reboot the servers

command**:** /sbin/reboot -t now

Another way to pass arguments to a module is using YAML syntax also called ‘complex args’

**-** name**:** restart webserver

service**:**

name**:** httpd

state**:** restarted

All modules return JSON format data. This means modules can be written in any programming language. Modules should be idempotent, and should avoid making any changes if they detect that the current state matches the desired final state. When used in an Ansible playbook, modules can trigger ‘change events’ in the form of notifying ‘handlers’ to run additional tasks.

Documentation for each module can be accessed from the command line with the ansible-doc tool:

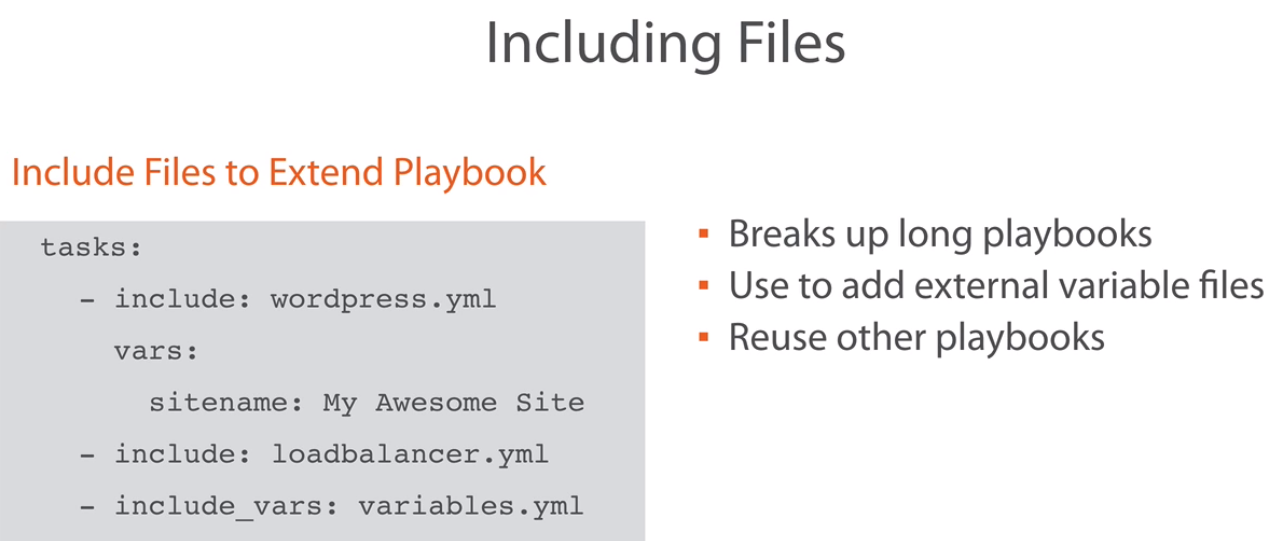
ansible-doc yum

For a list of all available modules, see the [Module Docs](https://docs.ansible.com/ansible/latest/modules/modules_by_category.html#modules-by-category), or run the following at a command prompt:

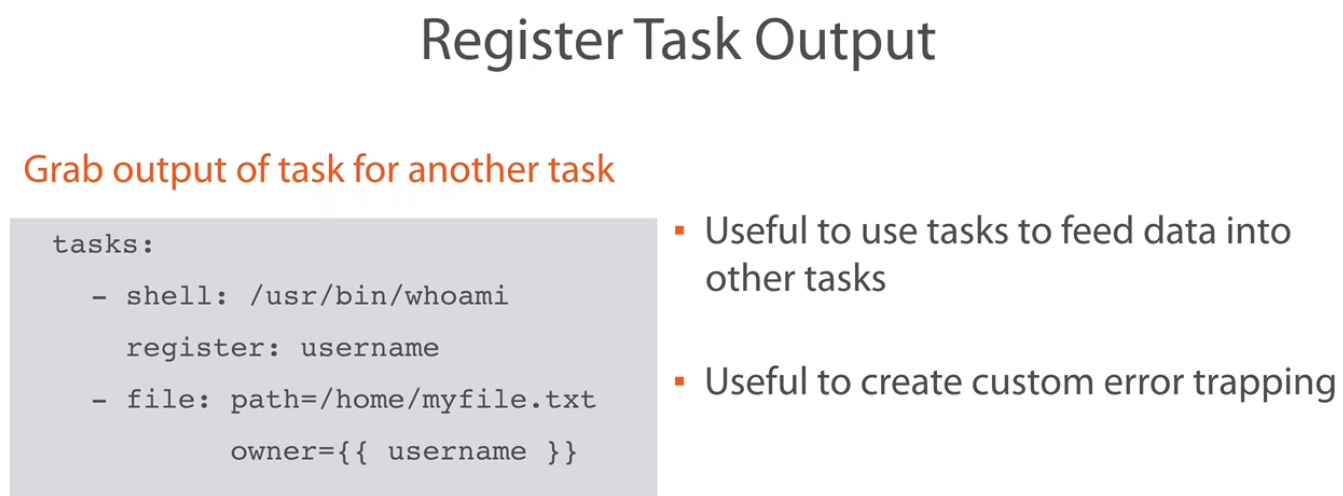
ansible-doc -l

**Playbooks and its feature:**

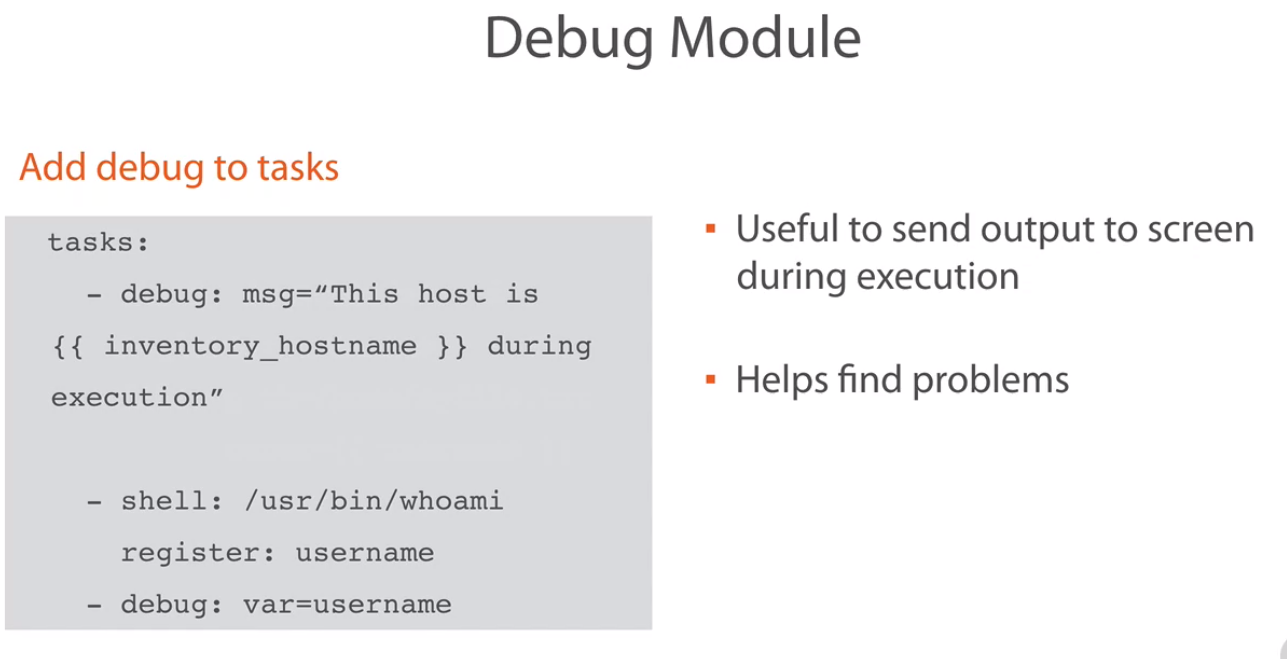
1. Include: include feature can be used to add another YAML or conf file in your current playbook. For example:



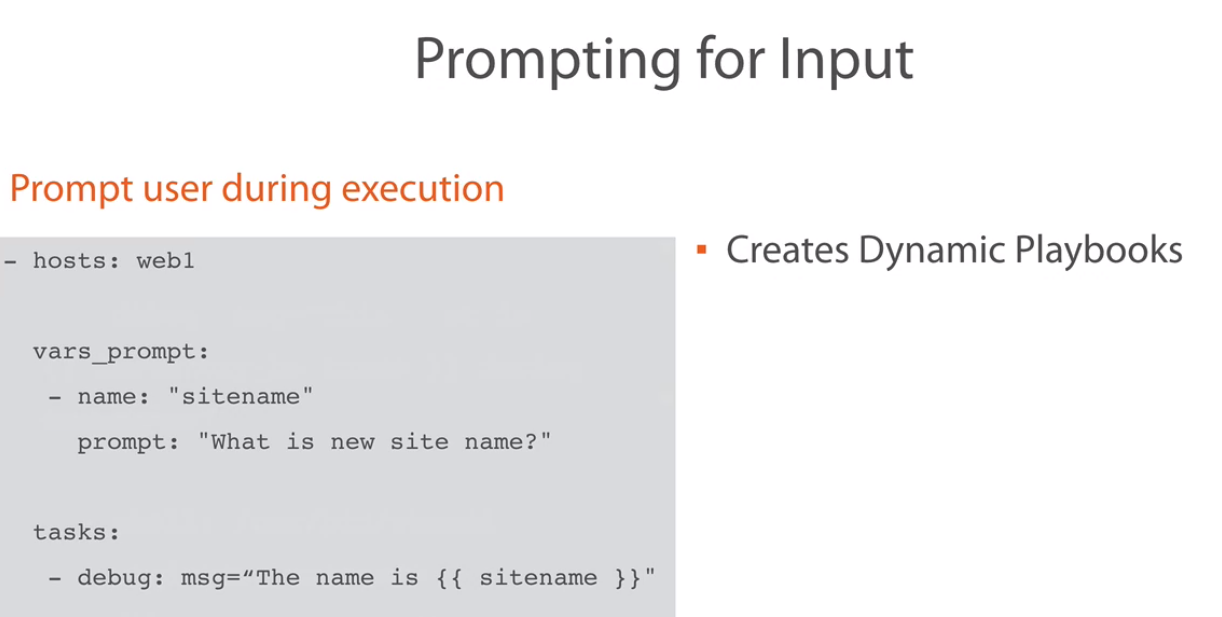
1. Register: this keyword is use to capture the output of a task, which can later be used in a subsequent task. Example:



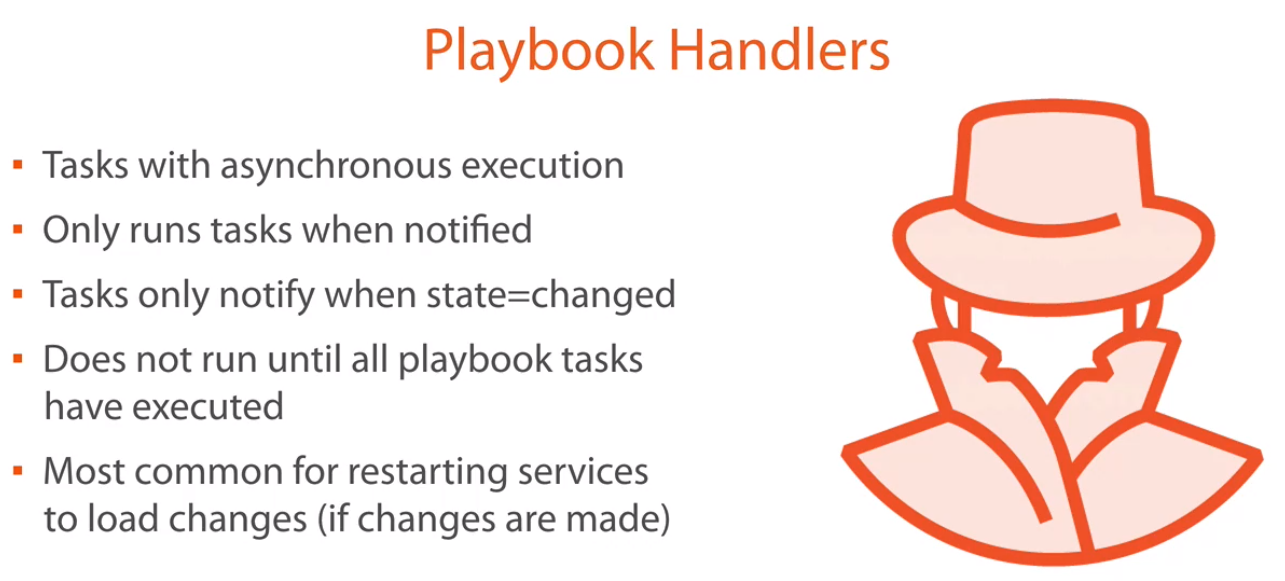
1. Debug Module: This module can be used to send the output to the screen during the execution, which can help user to identify the problem:



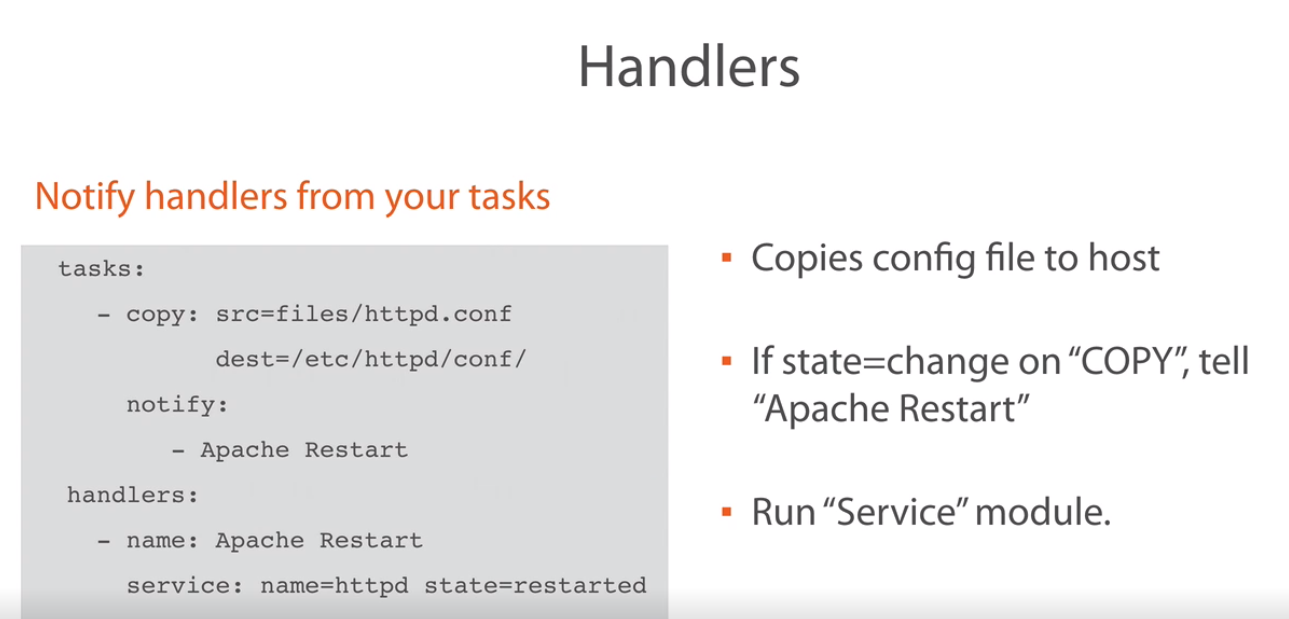
1. Prompting for input: This is used to create dynamic playbook. Point to note is that the alphabet won’t be visible when you type them.

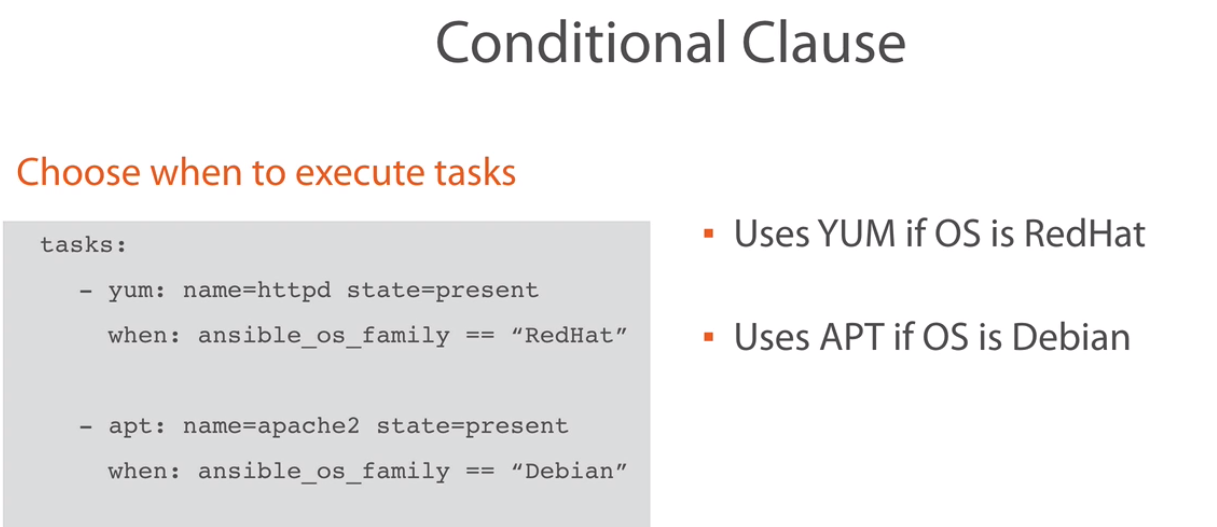
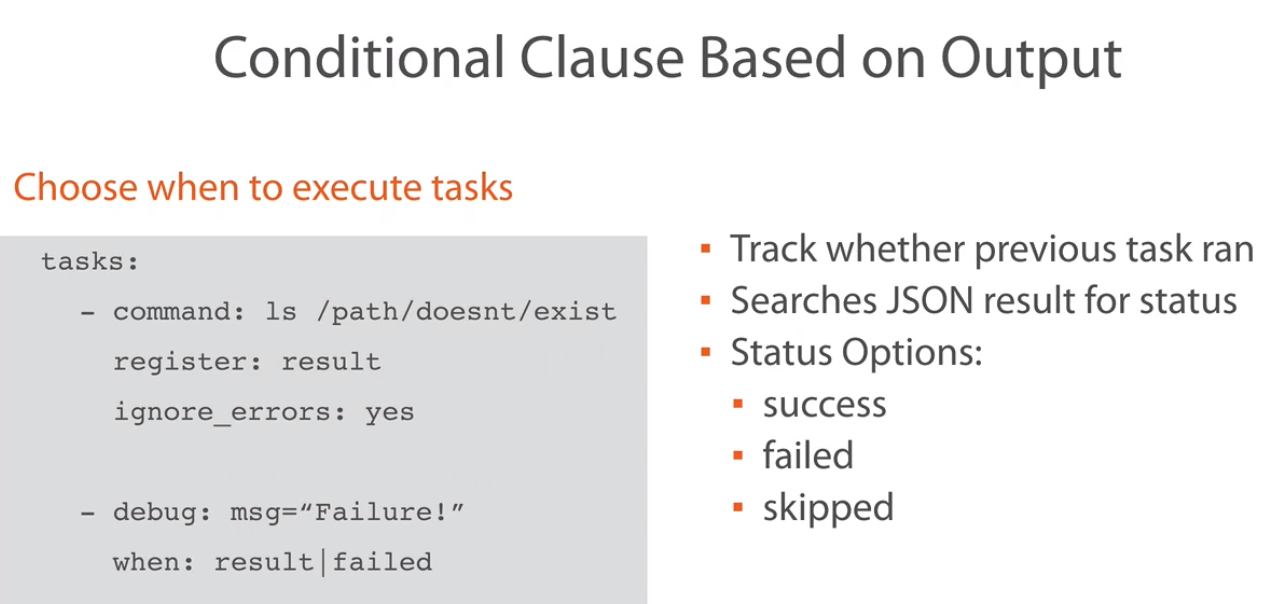


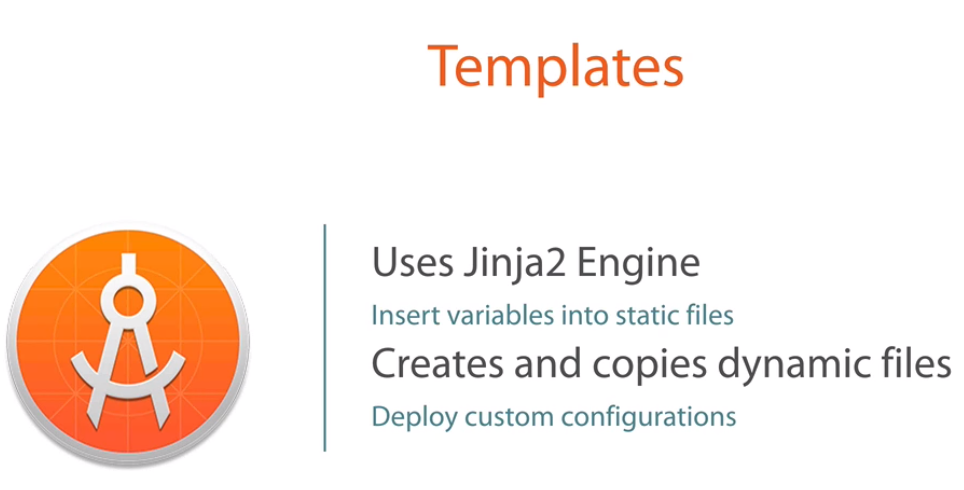
1. Playbook handlers: Can be used for various purposes as mentioned below:

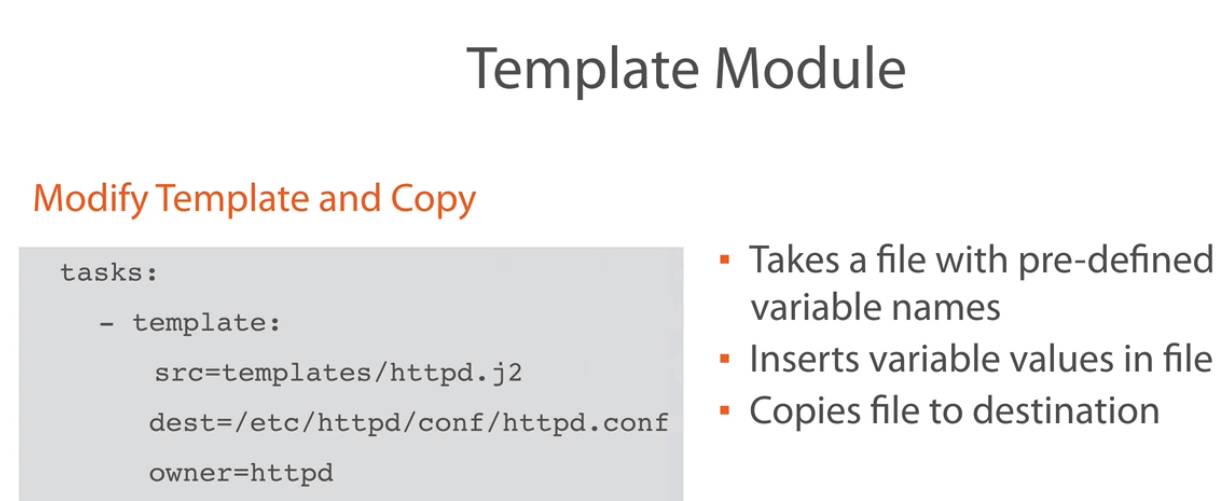


Handlers are used using the ‘notify’ keyword as shown below:



1. Conditional clause: these clause helps in deciding a condition, examples are like below:  
     
     
     
   

7. Templates:   
  




Ansible- Role:-

Ansible galaxy: a place to download the already created roles by several other users:

Basic command to install a role:



