Table of Contents

Working with playbooks

- Ad hoc commands can run a single, simple task against a set of targeted hosts as a one-time command.
- The real power of Ansible, however, is in learning how to use playbooks to run multiple, complex tasks against a set of targeted hosts in an easily repeatable manner.
- A play is an ordered set of tasks which should be run against hosts selected from your `inventory`.
- A playbook is a text file that contains a list of one or more plays to run in order.

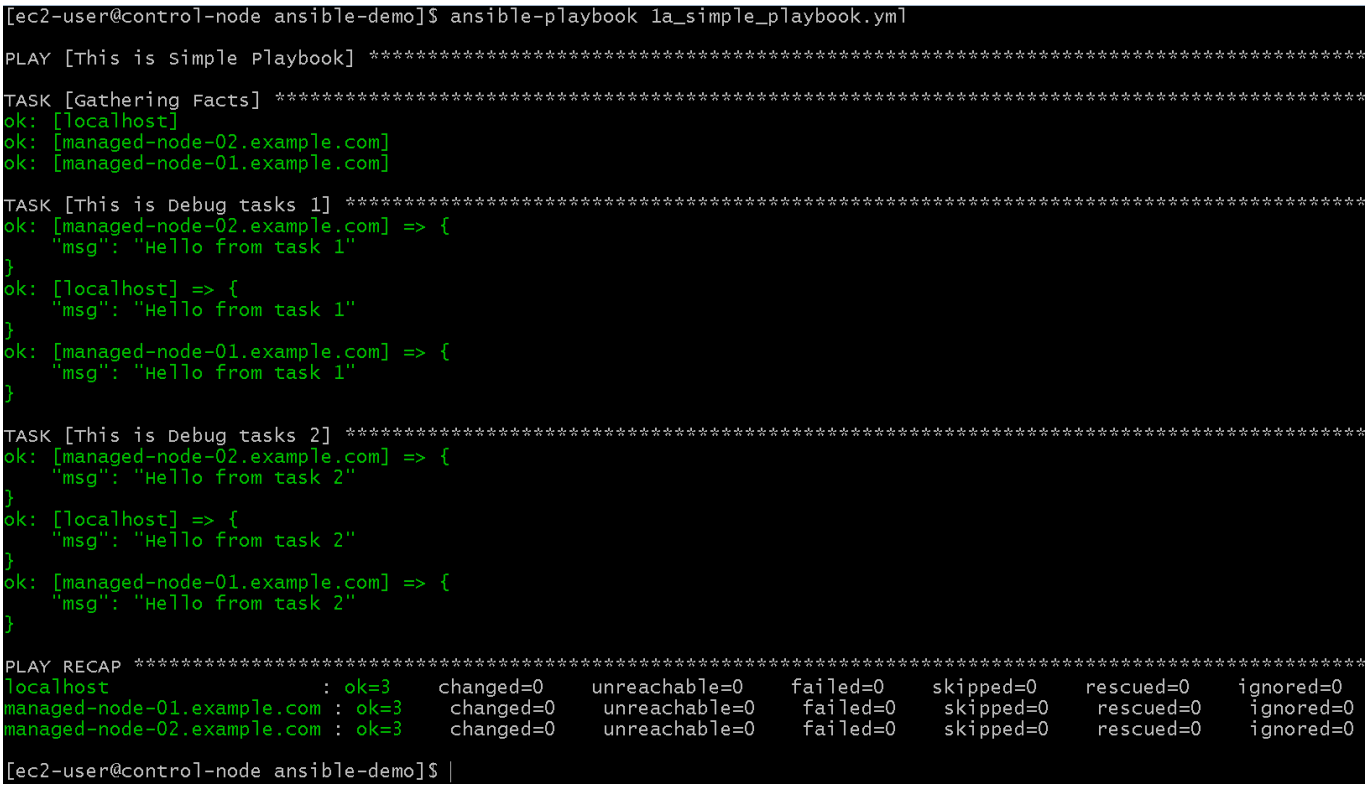
> In simple words :
> *`Playbook`* ***contains*** *`Plays`*
> *`Plays`* ***contains*** *`tasks`*
> *`Task`* ***runs*** *`modules`*.

Plays allow you to change a lengthy, complex set of manual administrative tasks into an easily repeatable routine with predictable and successful outcomes. In a playbook, you can save the sequence of tasks in a play into a human-readable and immediately runnable form.

Executing Ansible Playbooks

- Ansible Playbook execution

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



- Syntax Verification

```
ansible-playbook --syntax-check main.yml
```

- Executing a Dry Run

```
ansible-playbook 1b_dry_run.yml --check
```

> View the .yml files and execute with `ansible-playbook <playbook.yml>`

- Magic Variables
  - `hostvars`
  - Contains the variables for managed hosts, and can be used to get the values for another managed host's variables. It won't include the managed host's facts if they haven't been gathered yet for that host.
  - `group_names`
  - Lists all groups the current managed host is in.
  - `groups`
  - Lists all groups and hosts in the inventory.
  - `inventory_hostname`
  - Contains the hostname for the current managed host as configured in the inventory. This may be different from the hostname reported by facts for various reasons.
  - To view more magic variables that are associated with a particular hosts, use below command:

```
ansible localhost -m debug -a 'var=hostvars[inventory_hostname]'
```

Adding a new managed node

- Launch another `centos` machine and perform below tasks
- set password for centos user, change the `/etc/ssh/sshd_config` for `passwordauth` to yes
- Do a `ssh-copy-id` to centos machine OR Append the public ssh key of ec2-user from control machine to centos users `~/.ssh/authorized_keys` file.
- Change the hostname of this **centos** machine and add it in the host variable

```
sudo hostnamectl set-hostname managed-node-03.example.com
```

- Add below line in the `inventory` file

```
[centos]
managed-node-03.example.com     ansible_user=centos
```

> You can define the user that Ansible uses to connect to remote devices as a variable with **ansible_user**, in a configuration file with **DEFAULT_REMOTE_USER**, as a command-line option with **-u**, and with the playbook keyword **remote_user**.

- Try `ping` module on newly launched centos node.

```
ansible centos -m ping
ansible centos -m user -a 'name=new_user uid=4000 state=present'
```

- Find the modules for any tasks

```
ansible-doc yum
```

Ansible Jinja Templates

- Ansible uses **Jinja2** templating to enable dynamic expressions and access to variables.
- A Jinja2 template is composed of multiple elements: data, variables and expressions.
- Those `variables and expressions` are replaced with their values when the Jinja2 template is rendered.
- The variables used in the template can be specified in the `vars` section of the playbook.

Where the templating happens?

- This all happens on your control machine before the task is sent and executed on the target machine.
- We don't need jinja2 packages on managed nodes.
- Only the information required by each task will be sent to remote nodes after template parsing.

Jinja2 Template Architecture

- Template files ends with `.j2` extension.
- A .j2 Jinja2 template file contains:
  - **{{ }}** : Used for embedding variables which prints their actual value during code execution.
  - **{% %}** : Used for control statements such as loops and if-else statements.
  - **{# #}** : To specify the comments.

Ansible template module

- Ansible's `template` module transfers template files to remote hosts while playbook execution. It works similarly to the copy module.
- The following example shows how to create a template with variables using two of the facts retrieved by Ansible from managed hosts: `ansible_hostname` and `ansible_date_time.date`.
- When the associated playbook is executed, those two facts will be replaced by their values in the managed host being configured.

> Check jinja-test ansible playbooks