

Ansible modules

Getting started

Emilien Macchi
Software Developer



ANSIBLE

emilien@redhat.com

- OpenStack Contributor since 2011
- Member of OpenStack Engineering group at Red Hat Canada
- Technical lead for the OpenStack installer
- Help converted our OpenStack Deployment Framework to use Ansible



Red Hat



openstack®

Agenda

- Overview of Ansible Plugins
- Action Action Plugins
- Ansible Modules
- Actions Plugins vs Modules
- Live programming

Overview of Ansible Plugins


- Pieces of Python code that augment Ansible's functionalities
- Ansible is shipped with handy plugins but we can write our own
- Different types of plugins: *action* (covered later), *cache*, *callback* (e.g. [ARA has one](#)), *connection*, *inventory*, *lookup*, *strategy*, *filters* (to extend what Jinja2 offers), etc
- Usually placed in ***/usr/share/ansible/plugins***
- <https://docs.ansible.com/ansible/latest/plugins/plugins.html>

Ansible Action Plugins

- Described as “*special type of Module*”
- Run on the “master” node (from where Ansible is executed, not the target)
- Can call multiple modules with `_execute_module()`
- Useful to reduce the number of task in a playbook
 - e.g. playbook with 10 tasks can be replaced by an action plugin (one task), calling the necessary modules.
 - Execution much faster (one task vs 10)
 - Can be very useful if a module is called multiple times with different inputs, its execution would be looped
- Usually placed in `/usr/share/ansible/plugins/action`
- <https://docs.ansible.com/ansible/latest/plugins/action.html>

```
def _delete_service(self, name, task_vars):  
    """Stop and disable a systemd service.  
  
    :param name: String for service name to stop and disable.  
    :param task_vars: Dictionary of Ansible task variables.  
    """  
    tvars = copy.deepcopy(task_vars)  
    results = self._execute_module(  
        module_name='systemd',  
        module_args=dict(state='stopped',  
                           name='tripleo_{}_healthcheck.timer'.format(name),  
                           enabled=False,  
                           daemon_reload=False),  
        task_vars=tvars  
    )  
    return results
```

Execute "systemd" module



Ansible Modules

- Reusable script that Ansible runs either locally or remotely on a target host
- Interact with the local machine with Python
- Defined interface, takes arguments, returns data
- Example of well-known modules: *command*, *copy*, *file*, *systemd*, etc
- Write your own for your needs
- Help with scalability
 - Less tasks
 - Executed faster than tasks
 - e.g. One module to replace 10 tasks will run much faster
- Usually placed in */usr/share/ansible/plugins/modules*

Action Plugins vs Modules

	Action plugin	Module
Where does it run?	On the executor, so be careful with CPU / memory if too many action plugins	On the target node (can also be run locally with <code>delegate_to: localhost</code>)
Dependencies on target	N/A	Python + modules required to run the Ansible module
Access to variables	Yes	No access. Can only reach facts of the target machine.
Use cases	<ul style="list-style-type: none">• Reduce Ansible tasks• prepare a node before executing a module	<ul style="list-style-type: none">• Reduce Ansible tasks• Use Python to interact with the target system

Live programming a Module

- Module code: <https://gitlab.com/-/snippets/2019902>
- Demo: <https://asciinema.org/a/vZVOIcWEc9sGfY5c2Bi59HwZA>

- Let me share my screen!

Thank you!

