Developing Modules

Topics

Welcome

This section discusses how to develop, debug, review, and test modules.

Ansible modules are reusable, standalone scripts that can be used by the Ansible API, or by the **ansible** or **ansible-playbook** programs. They return information to ansible by printing a JSON string to stdout before exiting. They take arguments in one of several ways which we'll go into as we work through this tutorial.

See About Modules (../modules.html) for a list of existing modules.

Modules can be written in any language and are found in the path specified by `ANSIBLE_LIBRARY` (../config.html#envvar-ANSIBLE_LIBRARY) or the `--module-path` command line option or in the library section of the Ansible configuration file.

Should You Develop A Module?

Before diving into the work of creating a new module, you should think about whether you actually *should* develop a module. Ask the following questions:

1. Does a similar module already exist?

There are a lot of existing modules available, you should check out the list of existing modules at About Modules (../modules.html)

2. Has someone already worked on a similar Pull Request?

It's possible that someone has already started developing a similar PR. There are a few ways to find open module Pull Requests:

- GitHub new module PRs
- All updates to modules
- New module PRs listed by directory search for *lib/ansible/modules/*

If you find an existing PR that looks like it addresses the issue you are trying to solve, please provide feedback on the PR - this will speed up getting the PR merged.

3. Should you use or develop an action plugin instead?

Action plugins get run on the master instead of on the target. For modules like file/copy/template, some of the work needs to be done on the master before the module executes on the target. Action plugins execute first on the master and can then execute the normal module on the target if necessary.

For more information about action plugins, read the action plugins documentation here.

4. Should you use a role instead?

Check out the roles documentation available here.

How To Develop A Module

The following topics will discuss how to develop and work with modules:

Ansible Module Development Walkthrough (developing_modules_general.html)
A general overview of how to develop, debug, and test modules.

Windows Ansible Module Development Walkthrough (developing_modules_general_windows.html)
A general overview of how to develop, debug and test Windows modules.

Documenting Your Module (developing_modules_documenting.html)
How to include in-line documentation in your module.

Conventions, Best Practices, and Pitfalls (developing_modules_best_practices.html)
Best practices, recommendations, and things to avoid.

Contributing Your Module to Ansible (developing_modules_checklist.html)
Checklist for contributing your module to Ansible.

Testing Ansible (testing.html)
Developing unit and integration tests.

Ansible and Python 3 (developing_python3.html)
Adding Python 3 support to modules (all new modules must be Python-2.6 and Python-3.5 compatible).

Information for submitting a group of modules (developing_modules_in_groups.html)
A guide for partners wanting to submit multiple modules.

See also

About Modules (../modules.html)
Learn about available modules

Developing Plugins (developing_plugins.html)
Learn about developing plugins

Python API (developing_api.html)
Learn about the Python API for playbook and task execution

GitHub modules directory
Browse module source code

Mailing List
Development mailing list

irc.freenode.net
#ansible IRC chat channel

Appendix: Module Utilities

Ansible provides a number of module utilities that provide helper functions that you can use when developing your own modules. The *basic.py* module utility provides the main entry point for accessing the Ansible library, and all Ansible modules must, at minimum, import from basic.py:

```
from ansible.module_utils.basic import *
```

The following is a list of module_utils files and a general description. The module utility source code lives in the *./lib/module_utils* directory under your main Ansible path - for more details on any specific module utility, please see the source code.

- a10.py - Utilities used by the a10_server module to manage A10 Networks devices.
- aireos.py - Definitions and helper functions for modules that manage Cisco WLC devices.
- api.py - Adds shared support for generic API modules.
- aos.py - Module support utilities for managing Apstra AOS Server.
- aruba.py - Helper functions for modules working with Aruba networking devices.
- asa.py - Module support utilities for managing Cisco ASA network devices.
- azure_rm_common.py - Definitions and utilities for Microsoft Azure Resource Manager template deployments.
- basic.py - General definitions and helper utilities for Ansible modules.
- cloudstack.py - Utilities for CloudStack modules.
- database.py - Miscellaneous helper functions for PostGRES and MySQL
- docker_common.py - Definitions and helper utilities for modules working with Docker.
- ec2.py - Definitions and utilities for modules working with Amazon EC2
- eos.py - Helper functions for modules working with EOS networking devices.
- f5.py - Helper functions for modules working with F5 networking devices.
- facts.py - Helper functions for modules that return facts.
- gce.py - Definitions and helper functions for modules that work with Google Compute Engine resources.
- ios.py - Definitions and helper functions for modules that manage Cisco IOS networking devices
- iosxr.py - Definitions and helper functions for modules that manage Cisco IOS-XR networking devices

- ismount.py - Contains single helper function that fixes os.path.ismount
- junos.py - Definitions and helper functions for modules that manage Junos networking devices
- known_hosts.py - utilities for working with known_hosts file
- manageiq.py - Functions and utilities for modules that work with ManageIQ platform and its resources.
- mysql.py - Allows modules to connect to a MySQL instance
- netapp.py - Functions and utilities for modules that work with the NetApp storage platforms.
- netcfg.py - Configuration utility functions for use by networking modules
- netcmd.py - Defines commands and comparison operators for use in networking modules
- netscaler.py - Utilities specifically for the netscaler network modules.
- network.py - Functions for running commands on networking devices
- nxos.py - Contains definitions and helper functions specific to Cisco NXOS networking devices
- openstack.py - Utilities for modules that work with Openstack instances.
- openswitch.py - Definitions and helper functions for modules that manage OpenSwitch devices
- powershell.ps1 - Utilities for working with Microsoft Windows clients
- pure.py - Functions and utilities for modules that work with the Pure Storage storage platforms.
- pycompat24.py - Exception workaround for Python 2.4.
- rax.py - Definitions and helper functions for modules that work with Rackspace resources.
- redhat.py - Functions for modules that manage Red Hat Network registration and subscriptions
- service.py - Contains utilities to enable modules to work with Linux services (placeholder, not in use).
- shell.py - Functions to allow modules to create shells and work with shell commands
- six/__init__.py - Bundled copy of the Six Python library to aid in writing code compatible with both Python 2 and Python 3.
- splitter.py - String splitting and manipulation utilities for working with Jinja2 templates
- urls.py - Utilities for working with http and https requests
- vca.py - Contains utilities for modules that work with VMware vCloud Air
- vmware.py - Contains utilities for modules that work with VMware vSphere VMs
- vyos.py - Definitions and functions for working with VyOS networking

Previous (overview_architecture.html)

Next (developing_modules_general.html)