Ansible module development

What, why and how

About

Jurica Zrna

Open Source technical sales representative – CEE

- Linux System Engineer
- Automation
- Cloud

jurica.zrna@ibm.com

What are modules

- Basic building blocks of Ansible
- Pieces of code executed on managed nodes
- Ideally idempotent

Module Types

- Action plugins
- Old style modules
- New style modules

Action plugins

- Look like modules
- Run on controller node
- Example:
  - Debug
- May invoke actual module to do some action on managed node
- Example:
  - template

Old style modules

- Expect a file with key-value pairs
- Reads the file and does work based on that
- Any module that can’t be identified by Ansible as new style is considered old style

New style modules

- Python
- Powershell
- JSONARGS modules
  - <<INCLUDE_ANSIBLE_MODULE_JSON_ARGS>>
- Non-native want JSON modules
  - WANT_JSON
- Binary modules

Why develop custom modules

- You need new functionality
- You have unique knowledge
- You want to improve Ansible

When not to

- Something similar exists
- You can use a role instead
- You actually need an Action plugin

The boilerplate

```python
#!/usr/bin/python

from ansible.module_utils.basic import AnsibleModule

def main():
  module = AnsibleModule(
    argument_spec=dict(
    ),
    supports_check_mode=False
  )

  module.exit_json(msg="Task done.")

if __name__ == '__main__':
    main()
```

argument_spec (1)

A dictionary that defines:

- supported arguments
- their type
- defaults
- more

argument_spec (2)

```
argument_spec = dict(
  optional_arg = dict(),
  required_arg = dict(required=True),
  secret_arg = dict(no_log=True),
  str_arg = dict(type='str'),
  int_list_arg = dict(type='list', elements='int'),
  default_arg = dict(default='value'),
  aliased_arg = dict(aliases=['alt_arg1', 'alt_arg2']),
  choice_arg = dict(choices=['option1', 'option2', 'option3']),
  failback_arg = dict(failback=(env_failback, ['ENV_VARIABLE']))
)
```

argument_spec (3)

```
argument_spec = dict(
  optional_arg1 = dict(),
  optional_arg2 = dict(),
  conditional_arg = dict(type='bool'),
),
mutually_exclusive = [
  ['optional_arg1', 'optional_arg2']
],
required_together = [
  ['optional_arg1', 'optional_arg2']
],
required_one_of = [
  ['optional_arg1', 'optional_arg2']
],
required_if = [
  ['conditional_arg', True, ['optional_arg1', 'optional_arg2']],
  ['conditional_arg', False, ['optional_arg1']],
```

Retrieving values

```
#!/usr/bin/python

from ansible.module_utils.basic import AnsibleModule

def main():
  module = AnsibleModule(
    argument_spec=dict(
      arg = dict()
    ),
    supports_check_mode=False
  )

  arg = module.params['arg']

  module.exit_json(msg="Task done.")

if __name__ == '__main__':
    main()
```

Exiting module

```
module.exit_json(data=return_data)

module.fail_json(msg=fail_message)
```

Check mode

```python
if module.check_mode:
  # report stuff to be done
  # ...
```

Documentation

Module documentation consists of:

- metadata
- documentation
- examples
- returns

Demo

Questions?

Links

- Ansible documentation:
  - https://docs.ansible.com/ansible/latest/dev_guide/developing_modules_general.html
- Demo code:
  - https://github.ibm.com/Jurica-Zrna/ansible-module-dev