
cisco
You make possible

How to Write an Ansible Module

John McDonough – DevNet Developer Advocate
@johnamcdonough

DEVNET-2215

cisco Live!
Barcelona | January 27–31, 2020

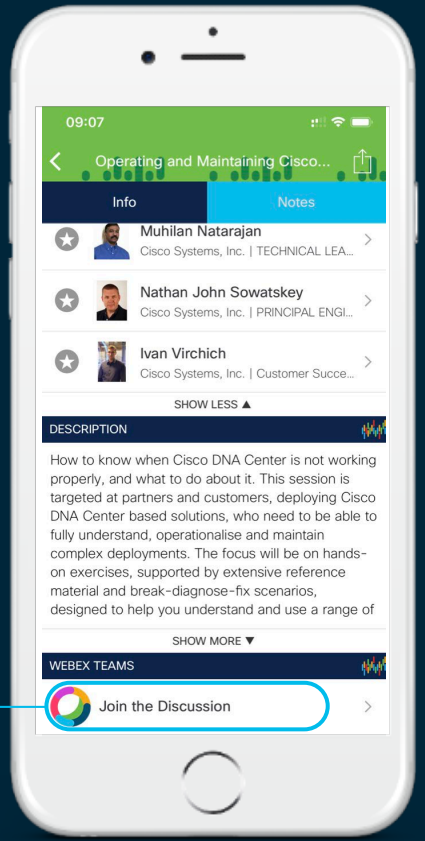
Cisco Webex Teams

Questions?

Use Cisco Webex Teams to chat with the speaker after the session

How

1. Find this session in the Cisco Events Mobile App
2. Click “Join the Discussion”
3. Install Webex Teams or go directly to the team space
4. Enter messages/questions in the team space



Learn more about the new DevNet Certifications and how you can prepare now!

	Associate Level	Specialist Level	Professional Level	Expert Level
Engineering				
Software				

Future Offering

Start Here | Upcoming Cisco DevNet Certifications

- Start at **Meet DevNet**

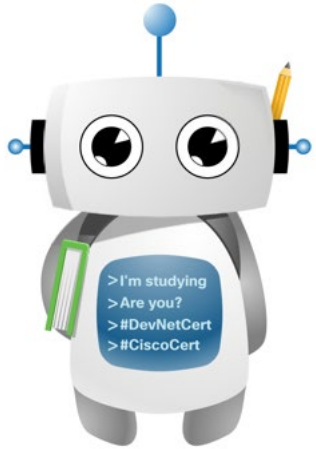
  DEVNET-2864: Getting ready for Cisco DevNet Certifications
  Offered daily at 9am, 1pm & 4pm at Meet DevNet

- Attend a **brownbag session**

  DEVNET-4099: DevNet Certifications: Bringing software practices & software skills to networking
  Offered daily 12:15-12:45 in the DevNet Zone Theater

- Visit the **Learning@Cisco** booth

- Scan this code to **sign up** for the latest updates or go to http://cs.co/20eur02



cisco Live!

Agenda

- DEVNET-2215 - How to Write an Ansible Module
- Ansible Module Basics
- The Parts
  - The Parts of an Ansible Module
- The Tests
  - Test the possibilities of an Ansible Module
- Conclusion

Introduction

Introduction – How to Write an Ansible Module

- My Environment
  - VS Code
    - Python Extension
  - Python 3
    - Virtual Environment
    - Activate
    - Install Ansible & UCS Python SDK
  - UCS Manager
- Code is at https://github.com/movinalot/DEVNET-2215

```
$ python3 -m venv venv
$ . venv/bin/activate
$ pip install ansible
$ pip install ucsmsdk

Or

$ pip install –r requirements.txt
```

Ansible Module Basics

- Single Logical Object operation
- Self-contained, Single file – Common elements can be shared
  - Common code in module_utils
  - Common documentation in doc_fragments
- AnsibleModule Python Class provides
  - Argument parsing and validation
  - JSON output formatting
- JSON in, JSON out

```
{ "ANSIBLE_MODULE_ARGS":
  {
    "name": "prod",
    ”descr": ”Production",
    "state": "present"
  }
}
```

The Parts

The Parts

1. import AnsibleModule
2. Get the parameters
3. Check the state based on the Parameters

   Does it exist? / Should it change? / Should it be deleted? - Idempotence
4. Check Mode

   Do it or Don't Do it
5. DOCUMENTATION / EXAMPLES / RETURN
6. doc_fragments & module_utils
7. Tests

The Smallest Ansible Module

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

from ansible.module_utils.basic import AnsibleModule

def run_module():
    module = AnsibleModule(
        argument_spec=dict(
        )
    )

    result = dict(changed=False)

    module.exit_json(**result)

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

The Ansible Module Parameters

```python
from ansible.module_utils.basic import AnsibleModule

def run_module():
    module = AnsibleModule(
        argument_spec=dict(
            name=dict(),
            descr=dict(),
            state=dict()
        )
    )

    result = dict(changed=False)

    module.exit_json(**result)

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

The Ansible Module Parameters – arguments

```python
from ansible.module_utils.basic import AnsibleModule

def run_module():
    module = AnsibleModule(
        argument_spec=dict(
            hostname=dict(type='str', required=True),
            username=dict(type='str', default='admin'),
            password=dict(type='str', required=True, no_log=True),
            name=dict(type='str'),
            descr=dict(type='str'),
            state=dict(type='str', default='present', choices=['present', 'absent'])
        )
    )

    result = dict(changed=False)

    module.exit_json(**result)

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

https://docs.ansible.com/ansible/latest/dev_guide/developing_program_flow_modules.html#argument-spec

Non-Idempotence

```
def run_module():
    module = AnsibleModule(
        argument_spec=dict(
            hostname=dict(type='str', required=True),
            username=dict(type='str', default='admin'),
            password=dict(type='str', required=True, no_log=True),
            name=dict(type='str'),
            descr=dict(type='str'),
            state=dict(type='str', default='present', choices=['present', 'absent'])
        )
    )

    from ucsmsdk.ucshandle import UcsHandle
    from ucsmsdk.mometa.org.OrgOrg import OrgOrg
    handle = UcsHandle(module.params['hostname'], module.params['username'], module.params['password'])
    handle.login()
    ucs_mo = OrgOrg(parent_mo_or_dn='org-root', name=module.params['name'], descr=module.params['descr'])
    handle.add_mo(ucs_mo, modify_present=True)
    handle.commit()
    handle.logout()

    # TODO: Add delete object code
```

create / update / delete the object without respect to its current state

This example shows create / update

Idempotence

```
handle.login()
ucs_mo = handle.query_dn('org-root/org-' + module.params['name'])

# Determine state change
if ucs_mo:
    # Object exists, should it exist? has anything changed?
    if module.params['state'] == 'present':
        # Any Object properties not match, that is a change
        if not ucs_mo.check_prop_match(**kwargs):
            changed = True

# Object does not exist but should, that is a change
else:
    if module.params['state'] == 'present':
        changed = True

# Object exists but should not, that is a change
if ucs_mo and module.params['state'] == 'absent':
    changed = True
```

```
# Apply state
if changed:
    if module.params['state'] == 'absent':
        handle.remove_mo(ucs_mo)
    else:
        kwargs['parent_mo_or_dn'] = 'org-root'
        kwargs['name'] = module.params['name']
        if module.params['descr'] is not None:
            kwargs['descr'] = module.params['descr']

        ucs_mo = OrgOrg(**kwargs)
        handle.add_mo(ucs_mo, modify_present=True)
    handle.commit()
```

Create if present specified or
Update only what is different
Delete if absent specified

Check Mode --check

```
module = AnsibleModule(
    argument_spec=dict(
        hostname=dict(type='str', required=True),
        username=dict(type='str', default='admin'),
        password=dict(type='str', required=True, no_log=True),
        name=dict(type='str'),
        descr=dict(type='str'),
        state=dict(type='str', default='present', choices=['present', 'absent'])
    ),
    required_if=[
        ['state', 'present', ['name']],
    ],
    supports_check_mode=True
)
```

```
# Apply state if not check_mode
if changed and not module.check_mode:
```

DOCUMENTATION / EXAMPLES / RETURN

```
DOCUMENTATION = r'''
---
module: ucs_org

short_description: Manages UCS Organizations for UCS Manager

description:
    - Manages UCS Organizations for UCS Manager.
    - Examples can be used with the UCS Platform Emulator U(https://cs.co/ucspe).

options:
    hostname:
        description:
        - IP address or hostname of Cisco UCS Manager.
        type: str
        required: yes
RETURN = r''' '''
```

```
EXAMPLES = r'''
- name: Add UCS Organization
  ucs_org_6:
    hostname: "{{ ucs_hostname }}"
    username: "{{ ucs_username }}"
    password: "{{ ucs_password }}"
    name: test
    description: testing org
    state: present

- name: Update UCS Organization
  ucs_org_6:
    hostname: "{{ ucs_hostname }}"
    username: "{{ ucs_username }}"
    password: "{{ ucs_password }}"
    name: test
    description: TESTING ORG
    state: present
```

doc_fragments / module_utils

```
class ModuleDocFragment(object):
    # Cisco UCS doc fragment
    DOCUMENTATION = '''
options:
  hostname:
    description:
    - IP address or hostname of Cisco UCS Manager.
    type: str
    required: yes
  username:
    description:
    - Username for Cisco UCS Manager authentication.
    type: str
    default: admin
  password:
    description:
    - Password for Cisco UCS Manager authentication.
    type: str
    required: yes
```

```
from ucsmsdk.ucshandle import UcsHandle
def ucs_login(hostname, username, password):
    """ Login to UCS """

    HANDLE = UcsHandle(hostname, username, password)
    HANDLE.login()
    return HANDLE
```

The Tests

Test the Module

- Test Creating
- Test Updating
- Test Deletion
- Test Check Mode
- Use Ansbile DSL
  - Asserts
  - Register
  - Comparisons

```yaml
# Setup (clean environment)
- name: org absent
  ucs_org_6: &org_absent
    <<: *login_info
    name: testorg
    state: absent


# Test present (check_mode)
- name: org present (check_mode)
  ucs_org_6: &org_present
    <<: *login_info
    name: testorg
  check_mode: yes
  register: cm_org_present


# Present (normal mode)
- name: org present (normal mode)
  ucs_org_6: *org_present
```

Conclusion

Got Questions? Come find me!

jomcdono@cisco.com

@johnamcdonough

http://github.com/movinalot

@CiscoDevNet

facebook.com/ciscodevnet

http://github.com/CiscoDevNet

Complete your online session survey

- Please complete your session survey after each session. Your feedback is very important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (starting on Thursday) to receive your Cisco Live t-shirt.
- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Content Catalog on ciscolive.com/emea.

Cisco Live sessions will be available for viewing on demand after the event at ciscolive.com.

Continue your education

Demos in the Cisco Showcase

Walk-In Labs

Meet the Engineer 1:1 meetings

Related sessions

Thank you

You make **possible**