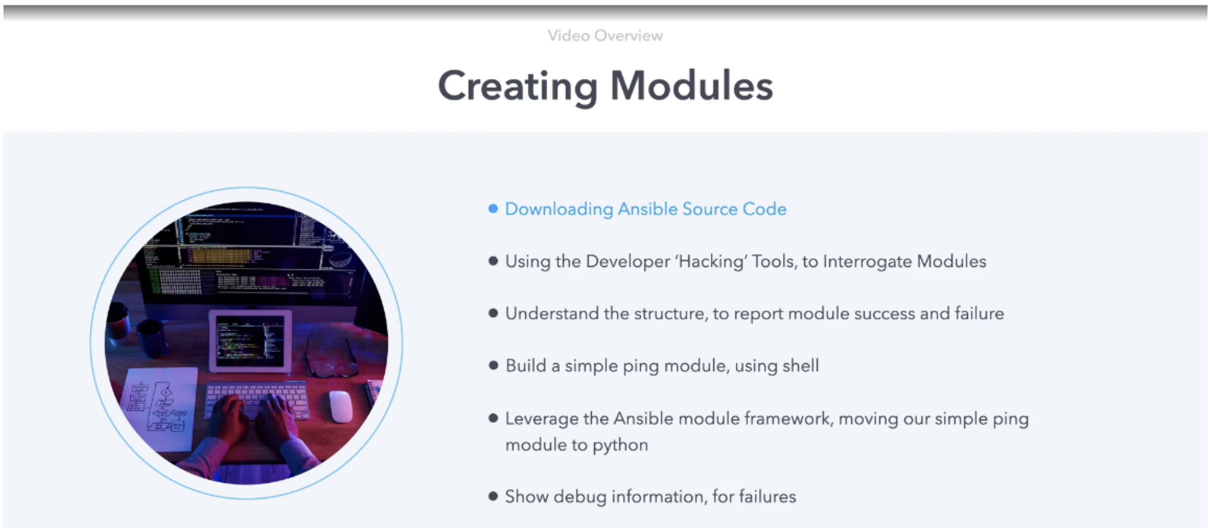
Chapter 8: Creating Modules and Plug-ins

1. Creating Modules



Video Overview

Creating Modules

- Downloading Ansible Source Code
- Using the Developer 'Hacking' Tools, to Interrogate Modules
- Understand the structure, to report module success and failure
- Build a simple ping module, using shell
- Leverage the Ansible module framework, moving our simple ping module to python
- Show debug information, for failures

1-1. Using the Developer 'Hacking' Tools

```
$ cd ~
$ git clone https://github.com/ansible/ansible.git

# ansible/hacking/test-module 에서 Python 경로 수정 후 저장

$ cd -
$ ~/ansible/hacking/test-module -m ~/ansible/lib/ansible/modules/command.py -a hostname
* including generated source, if any, saving to: /home/ansible/.ansible_module_generated
* ansiballz module detected; extracted module source to: /home/ansible/debug_dir
***********************************
RAW OUTPUT

{"changed": true, "stdout": "ubuntu-c", "stderr": "", "rc": 0, "cmd": ["hostname"], "start": "2022-07-03 11:11:57.886569", "end": "202


***********************************
PARSED OUTPUT
{
    "changed": true,
    "cmd": [
        "hostname"
    ],
    "delta": "0:00:00.002278",
    "end": "2022-07-03 11:11:57.888847",
    "invocation": {
        "module_args": {
            "_raw_params": "hostname",
            "_uses_shell": false,
            "argv": null,
            "chdir": null,
            "creates": null,
            "executable": null,
            "removes": null,
            "stdin": null,
            "stdin_add_newline": true,
            "strip_empty_ends": true
        }
    },
    "msg": "",
    "rc": 0,
    "start": "2022-07-03 11:11:57.886569",
    "stderr": "",
    "stdout": "ubuntu-c"
}

$ ~/ansible/hacking/test-module -m ~/ansible/lib/ansible/modules/command.py -a xyz
* including generated source, if any, saving to: /home/ansible/.ansible_module_generated
* ansiballz module detected; extracted module source to: /home/ansible/debug_dir
***********************************
RAW OUTPUT

{"rc": 2, "stdout": "", "stderr": "", "cmd": "xyz", "failed": true, "msg": "[Errno 2] No such file or directory: b'xyz'", "invocation"


***********************************
PARSED OUTPUT
{
    "cmd": "xyz",
    "failed": true,
    "invocation": {
        "module_args": {
            "_raw_params": "xyz",
            "_uses_shell": false,
            "argv": null,
            "chdir": null,
            "creates": null,
            "executable": null,
            "removes": null,
            "stdin": null,
            "stdin_add_newline": true,
            "strip_empty_ends": true
        }
    },
    "msg": "[Errno 2] No such file or directory: b'xyz'",
    "rc": 2,
    "stderr": "",
    "stdout": ""
}
```

test-module 에 전달되는 `-a <argument>` 에서 argument는 실행가능한 스크립트(any language)임을 알 수 있다. 또한, 스크립트의 최종 출력결과는 json 포맷이어야 한다.

```
#!/bin/bash

ping -c 1 127.0.0.1 >/dev/null 2>/dev/null

if [ $? == 0 ];
  then
  echo "{\"changed\": true, \"rc\": 0}"
else
  echo "{\"failed\": true, \"msg\": \"failed to ping\", \"rc\": 1}"
fi
```

```
$ ./icmp.sh
{"changed": true, "rc": 0}

$ cd ../02
$ ./icmp.sh
{"failed": true, "msg": "failed to ping", "rc": 1}
```

hacking-module 을 통해 사용자 스크립트 모듈 사용하기

```
$ cd ../03
$ ~/ansible/hacking/test-module -m icmp.sh
* including generated source, if any, saving to: /home/ansible/.ansible_module_generated
***********************************
RAW OUTPUT
{"changed": true, "rc": 0}


***********************************
PARSED OUTPUT
{
    "changed": true,
    "rc": 0
}
```

사용자 스크립트 모듈이 파라미터를 받아 처리할 수 있도록 모듈 수정

```
#!/bin/bash

# Capture inputs, these are passed as a file to the module
source $1 >/dev/null 2>&1

# Set our variables, set default if not assigned
TARGET=${target:-127.0.0.1}

ping -c 1 ${TARGET} >/dev/null 2>/dev/null

if [ $? == 0 ];
  then
  echo "{\"changed\": true, \"rc\": 0}"
else
  echo "{\"failed\": true, \"msg\": \"failed to ping\", \"rc\": 1}"
fi
```

```
$ cd ../04
$ ~/ansible/hacking/test-module -m icmp.sh
* including generated source, if any, saving to: /home/ansible/.ansible_module_generated
***********************************
RAW OUTPUT
{"changed": true, "rc": 0}


***********************************
PARSED OUTPUT
{
    "changed": true,
    "rc": 0
}

#######################################################################
#
# 파라미터 전달
#
#######################################################################

$ ~/ansible/hacking/test-module -m icmp.sh -a 'target=centos1'
* including generated source, if any, saving to: /home/ansible/.ansible_module_generated
***********************************
RAW OUTPUT
{"changed": true, "rc": 0}


***********************************
PARSED OUTPUT
{
    "changed": true,
    "rc": 0
}

$ ~/ansible/hacking/test-module -m icmp.sh -a 'target=centos4'
* including generated source, if any, saving to: /home/ansible/.ansible_module_generated
***********************************
RAW OUTPUT
{"failed": true, "msg": "failed to ping", "rc": 1}


***********************************
PARSED OUTPUT
{
    "failed": true,
    "msg": "failed to ping",
    "rc": 1
}

# 생성된 모듈 파일
$ cat /home/ansible/.ansible_module_generated
#!/bin/bash

# Capture inputs, these are passed as a file to the module
source $1 >/dev/null 2>&1

# Set our variables, set default if not assigned
TARGET=${target:-127.0.0.1}

ping -c 1 ${TARGET} >/dev/null 2>/dev/null

if [ $? == 0 ];
  then
  echo "{\"changed\": true, \"rc\": 0}"
else
```

```
  echo "{\"failed\": true, \"msg\": \"failed to ping\", \"rc\": 1}"
fi

# 모듈 파일에 전달된 파라미터
$ cat /home/ansible/.ansible_test_module_arguments
target=centos4

# 생성된 모듈과 파라미터를 이용해 테스트하기
$ /home/ansible/.ansible_module_generated /home/ansible/.ansible_test_module_arguments
{"failed": true, "msg": "failed to ping", "rc": 1}
```

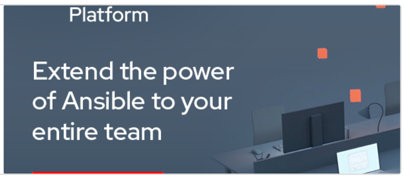
1-2. 사용자 모듈 라이브러리

```
$ cd ../05
$ ls -alh
total 16K
drwxr-xr-x 9 ansible ansible 288 Jun 30 23:24 .
drwxr-xr-x 9 ansible ansible 288 Jun 30 23:24 ..
-rwxr-xr-x 1 ansible ansible  63 Jun 30 23:24 ansible.cfg
drwxr-xr-x 4 ansible ansible 128 Jun 30 23:24 group_vars
drwxr-xr-x 4 ansible ansible 128 Jun 30 23:24 host_vars
-rwxr-xr-x 1 ansible ansible  95 Jun 30 23:24 hosts
-rwxr-xr-x 1 ansible ansible 523 Jun 30 23:24 icmp_fail_playbook.yaml
-rwxr-xr-x 1 ansible ansible 523 Jun 30 23:24 icmp_playbook.yaml
drwxr-xr-x 3 ansible ansible  96 Jun 30 23:24 library
$ ls library
icmp

# 확장자 .sh 가 없을 뿐 이전에 만든 icmp 스크립트와 동일하다.
$ cat library/icmp
#!/bin/bash

# Capture inputs, these are passed as a file to the module
source $1 >/dev/null 2>&1

# Set our variables, set default if not assigned
TARGET=${target:-127.0.0.1}

ping -c 1 ${TARGET} >/dev/null 2>/dev/null

if [ $? == 0 ];
  then
  echo "{\"changed\": true, \"rc\": 0}"
else
  echo "{\"failed\": true, \"msg\": \"failed to ping\", \"rc\": 1}"
fi

# 사용자가 만든 모듈(icmp)과 파라미터를 정의하여 플레이북으로 정의
$ cat icmp_playbook.yaml
---
# YAML documents begin with the document separator ---

# The minus in YAML this indicates a list item.  The playbook contains a list
# of plays, with each play being a dictionary
-

  # Hosts: where our play will run and options it will run with
  hosts: linux

  # Tasks: the list of tasks that will be executed within the play, this section
  # can also be used for pre and post tasks
  tasks:

    - name: Test icmp module
      icmp:
        target: 127.0.0.1

# Three dots indicate the end of a YAML document
...

# 실행(성공)
$ ansible-playbook icmp_playbook.yaml
# 실행(실패)
$ ansible-playbook icmp_fail_playbook.yaml
```

Developing modules - Ansible Documentation

A module is a reusable, standalone script that Ansible runs on your behalf, either locally or remotely. Modules interact with your local machine, an API, or a remote system to perform specific tasks like changing a database password or spinning up a cloud instance.

https://docs.ansible.com/ansible/latest/dev_guide/developing_modules_general.html



```
#!/usr/bin/python3

ANSIBLE_METADATA = {
    'metadata_version': '1.1',
    'status': ['preview'],
    'supported_by': 'community'
}

DOCUMENTATION = '''
---
module: icmp

short_description: simple module for icmp ping

version_added: "2.10"

description:
    - "simple module for icmp ping"

options:
    target:
        description:
            - The target to ping
        required: true

author:
    - James Spurin (@spurin)
'''

EXAMPLES = '''
# Ping an IP
- name: Ping an IP
  icmp:
    target: 127.0.0.1

# Ping a host
- name: Ping a host
  icmp:
    target: centos1
'''

RETURN = '''
'''

from ansible.module_utils.basic import AnsibleModule

def run_module():
    # define the available arguments/parameters that a user can pass to
    # the module
    module_args = dict(
        target=dict(type='str', required=True)
    )

    # seed the result dict in the object
    # we primarily care about changed and state
    # change is if this module effectively modified the target
    # state will include any data that you want your module to pass back
    # for consumption, for example, in a subsequent task
    result = dict(
        changed=False
    )

    # the AnsibleModule object will be our abstraction working with Ansible
    # this includes instantiation, a couple of common attr would be the
    # args/params passed to the execution, as well as if the module
    # supports check mode
    module = AnsibleModule(
        argument_spec=module_args,
        supports_check_mode=True
    )

    # if the user is working with this module in only check mode we do not
    # want to make any changes to the environment, just return the current
    # state with no modifications
    if module.check_mode:
        return result
    # manipulate or modify the state as needed (this is going to be the
    # part where your module will do what it needs to do)
    ping_result = module.run_command('ping -c 1 {}'.format(module.params['target']))

    # use whatever logic you need to determine whether or not this module
    # made any modifications to your target
    if module.params['target']:
        result['debug'] = ping_result
        result['rc'] = ping_result[0]
        if result['rc']:
          result['failed'] = True
          module.fail_json(msg='failed to ping', **result)
        else:
          result['changed'] = True
          module.exit_json(**result)

def main():
    run_module()

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

```
$ ~/ansible/hacking/test-module -m icmp.py -a 'target=127.0.0.1'
$ ~/ansible/hacking/test-module -m icmp.py -a 'target=128.0.0.1'
```

모듈 문서 보기

```
$ cd ../07
$ ansible-doc -M library icmp
> ICMP    (/home/ansible/diveintoansible/Creating Modules and Plugins/Creating Modules/07/library/icmp.py)

        simple module for icmp ping

ADDED IN: version 2.10

OPTIONS (= is mandatory):

= target
        The target to ping



AUTHOR: James Spurin (@spurin)

METADATA:
  metadata_version: '1.1'
  status:
  - preview
  supported_by: community


EXAMPLES:

# Ping an IP
- name: Ping an IP
  icmp:
    target: 127.0.0.1

# Ping a host
- name: Ping a host
  icmp:
    target: centos1
```

2. Creating Plug-Ins

Video Overview

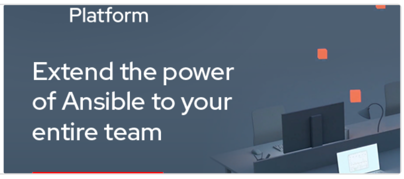
Creating Plugins

- Discuss the various types of Plugins
- Create a lookup_plugin
- Create a filter_plugin

Developing plugins - Ansible Documentation

Plugins augment Ansible's core functionality with logic and features that are accessible to all modules. Ansible collections include a number of handy plugins, and you can easily write your own. All plugins must: Once you've reviewed these general guidelines, you can skip to the particular type of plugin you want to

https://docs.ansible.com/ansible/latest/dev_guide/developing_plugins.html



ansible/items.py at devel · ansible/ansible

Ansible is a radically simple IT automation platform that makes your applications and systems easier to deploy and maintain. Automate everything from code deployment to network configuration to cloud management, in a language that approaches plain English, using SSH, with no agents to install on remote

https://github.com/ansible/ansible/blob/devel/lib/ansible/plugins/lookup/items.py

ansible/host_group_vars.py at devel · ansible/ansible

Ansible is a radically simple IT automation platform that makes your applications and systems easier to deploy and maintain. Automate everything from code deployment to network configuration to cloud management, in a language that approaches plain English, using SSH, with no agents to install on remote

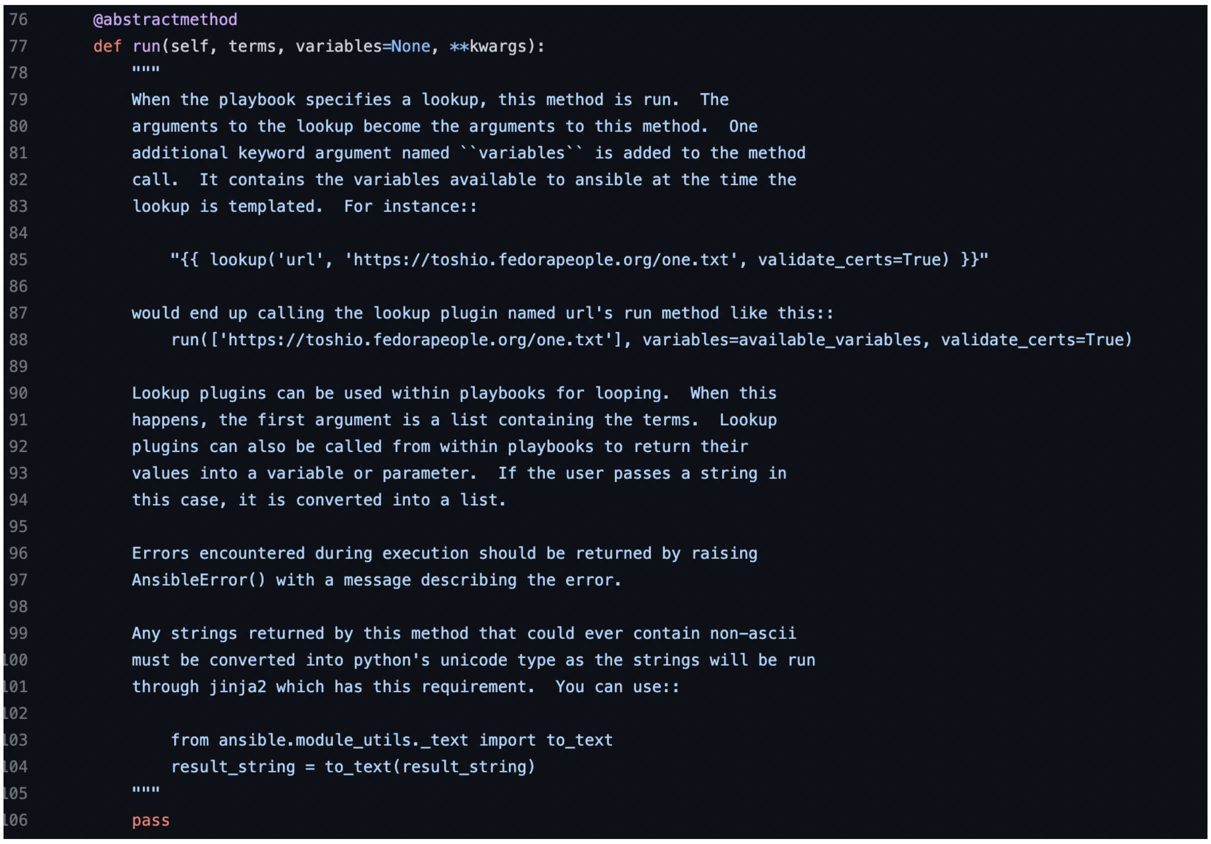
https://github.com/ansible/ansible/blob/devel/lib/ansible/plugins/vars/host_group_vars.py

Lookup 플러그인의 with_items에 정렬기능으로 변경

```
$ cd /home/ansible/diveintoansible/Creating Modules and Plugins/Creating Plugins/01

$ mkdir lookup_plugins

$ cd lookup_plugins/

$ wget https://raw.githubusercontent.com/ansible/ansible/devel/lib/ansible/plugins/lookup/items.py
--2022-07-03 12:25:42--  https://raw.githubusercontent.com/ansible/ansible/devel/lib/ansible/plugins/lookup/items.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1868 (1.8K) [text/plain]
Saving to: ‘items.py’

items.py                                     100%[==================================================================================

2022-07-03 12:25:43 (1.05 MB/s) - ‘items.py’ saved [1868/1868]

$ mv items.py sorted_items.py
```

```
from ansible.plugins.lookup import LookupBase
```

lookup 플러그인의 __init__.py 의 run 메서드. 아무런 일도 하지 않는 추상 메서드 이다. 따라서, 이 클래스를 상속받은 플러그인을 작성할때는 run 메서드를 구현해야 한다.



```
    @abstractmethod
    def run(self, terms, variables=None, **kwargs):
        """
        When the playbook specifies a lookup, this method is run.  The
        arguments to the lookup become the arguments to this method.  One
        additional keyword argument named ``variables`` is added to the method
        call.  It contains the variables available to ansible at the time the
        lookup is templated.  For instance::

            "{{ lookup('url', 'https://toshio.fedorapeople.org/one.txt', validate_certs=True) }}"

        would end up calling the lookup plugin named url's run method like this::
            run(['https://toshio.fedorapeople.org/one.txt'], variables=available_variables, validate_certs=True)

        Lookup plugins can be used within playbooks for looping.  When this
        happens, the first argument is a list containing the terms.  Lookup
        plugins can also be called from within playbooks to return their
        values into a variable or parameter.  If the user passes a string in
        this case, it is converted into a list.

        Errors encountered during execution should be returned by raising
        AnsibleError() with a message describing the error.

        Any strings returned by this method that could ever contain non-ascii
        must be converted into python's unicode type as the strings will be run
        through jinja2 which has this requirement.  You can use::

            from ansible.module_utils._text import to_text
            result_string = to_text(result_string)
        """
        pass
```

https://github.com/ansible/ansible/blob/devel/lib/ansible/plugins/lookup/__init__.py#L77

```
# (c) 2012, Michael DeHaan <michael.dehaan@gmail.com>
# (c) 2017 Ansible Project
# GNU General Public License v3.0+ (see COPYING or https://www.gnu.org/licenses/gpl-3.0.txt)
from __future__ import (absolute_import, division, print_function)
__metaclass__ = type

DOCUMENTATION = """
    name: items
    author: Michael DeHaan
    version_added: historical
    short_description: list of items
    description:
      - this lookup returns a list of items given to it, if any of the top level items is also a list it will flatten it, but it will
    notes:
      - this is the standard lookup used for loops in most examples
      - check out the 'flattened' lookup for recursive flattening
      - if you do not want flattening nor any other transformation look at the 'list' lookup.
    options:
      _terms:
        description: list of items
        required: True
"""

EXAMPLES = """
- name: "loop through list"
  ansible.builtin.debug:
    msg: "An item: {{ item }}"
  with_sorted_items:
    - 1
    - 2
    - 3

- name: add several users
  ansible.builtin.user:
    name: "{{ item }}"
    groups: "wheel"
    state: present
  with_sorted_items:
     - testuser1
     - testuser2

- name: "loop through list from a variable"
  ansible.builtin.debug:
    msg: "An item: {{ item }}"
  with_sorted_items: "{{ somelist }}"

- name: more complex items to add several users
  ansible.builtin.user:
    name: "{{ item.name }}"
    uid: "{{ item.uid }}"
    groups: "{{ item.groups }}"
    state: present
  with_sorted_items:
     - { name: testuser1, uid: 1002, groups: "wheel, staff" }
     - { name: testuser2, uid: 1003, groups: staff }

"""

RETURN = """
  _raw:
    description:
      - once flattened list
    type: list
"""

from ansible.plugins.lookup import LookupBase


class LookupModule(LookupBase):

    def run(self, terms, **kwargs):

        return self._flatten(sorted(terms, key=str))
```

```
---
# YAML documents begin with the document separator ---

# The minus in YAML this indicates a list item.  The playbook contains a list
# of plays, with each play being a dictionary
-

  # Hosts: where our play will run and options it will run with
  hosts: centos1

  # Tasks: the list of tasks that will be executed within the play, this section
  # can also be used for pre and post tasks
  tasks:

     - name: loop through list
       debug:
         msg: "An item: {{item}}"
       with_sorted_items:
         - 3
         - 2
         - 1
         - Z
         - A
         - M

# Three dots indicate the end of a YAML document
...
```

```
$ ansible-playbook sorted_items_playbook.yaml

PLAY [centos1] ***********************************************************************************************************************

TASK [Gathering Facts] ***************************************************************************************************************
ok: [centos1]

TASK [loop through list] *************************************************************************************************************
ok: [centos1] => (item=1) => {
    "msg": "An item: 1"
}
ok: [centos1] => (item=2) => {
    "msg": "An item: 2"
}
ok: [centos1] => (item=3) => {
    "msg": "An item: 3"
}
ok: [centos1] => (item=A) => {
    "msg": "An item: A"
}
ok: [centos1] => (item=M) => {
    "msg": "An item: M"
}
```

```
ok: [centos1] => (item=Z) => {
    "msg": "An item: Z"
}

PLAY RECAP **************************************************************************************************************************
centos1                    : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```