Custom Ansible Modules

Florin Lipan

What?

- „**Reusable, standalone scripts** that can be used by Ansible”
- „They return information to ansible by printing a **JSON** string to stdout before exiting”
- „They take **arguments** in one of several ways”

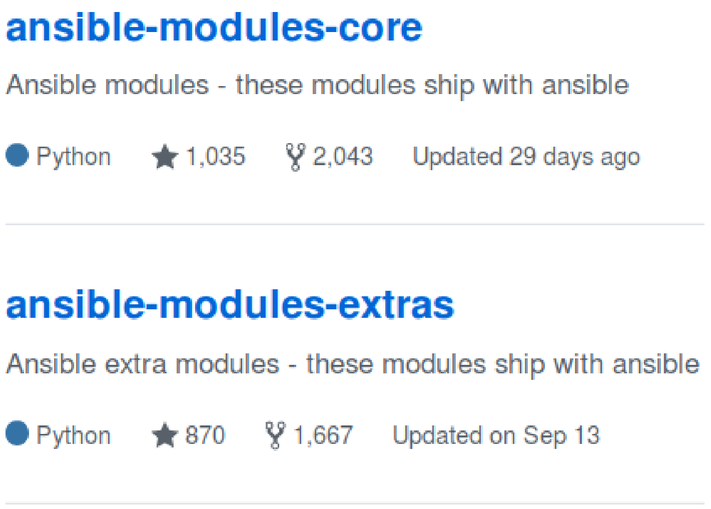
Why?

- **Services** that are not mapped to existing Ansible modules
- **Custom logic** that is not mapped to existing Ansible modules
- Interfacing with **other programming languages**
- **Hiding complexity** behind a simple interface

1378

Sa'im al-Dahr is hanged, for blowing the nose off the Sphinx.

1378



ansible-modules-core

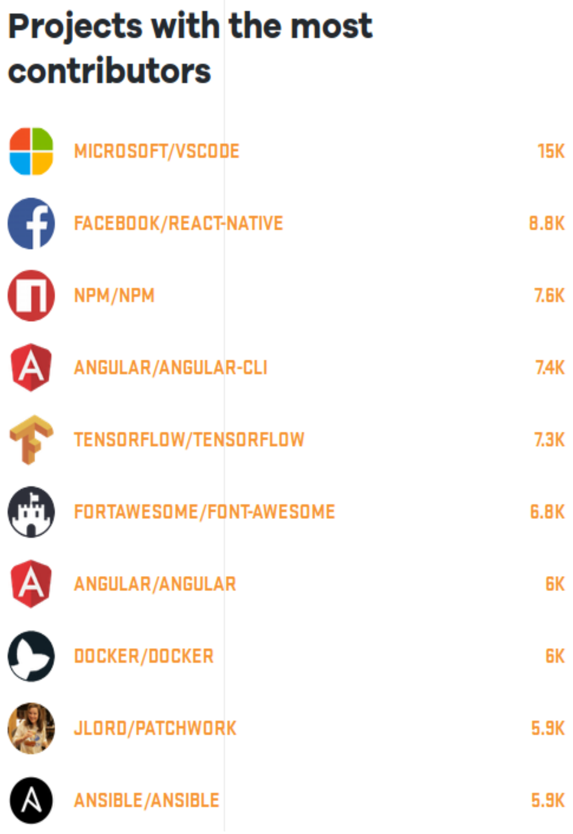
Ansible modules - these modules ship with ansible

Python ★ 1,035 2,043 Updated 29 days ago

ansible-modules-extras

Ansible extra modules - these modules ship with ansible

Python ★ 870 1,667 Updated on Sep 13



Projects with the most contributors

MICROSOFT/VSCODE	15K
FACEBOOK/REACT-NATIVE	8.8K
NPM/NPM	7.6K
ANGULAR/ANGULAR-CLI	7.4K
TENSORFLOW/TENSORFLOW	7.3K
FORTAWESOME/FONT-AWESOME	6.8K
ANGULAR/ANGULAR	6K
DOCKER/DOCKER	6K
JLORD/PATCHWORK	5.9K
ANSIBLE/ANSIBLE	5.9K

Why not?

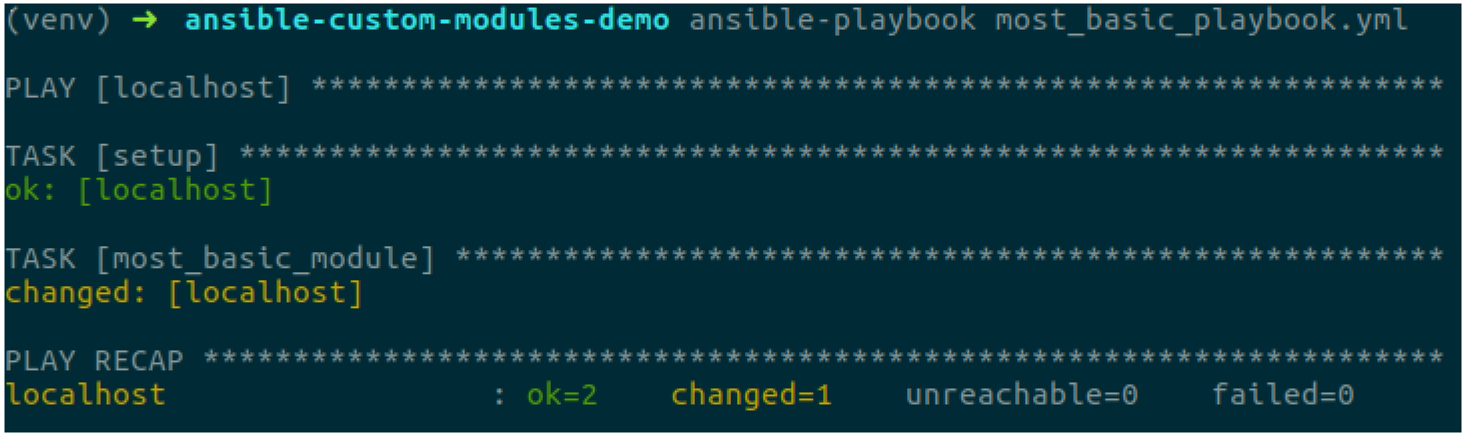
- Consider **roles**
- Consider action/vars/lookup/inventory **plugins**

Most basic module

```
# library/most_basic_module.py

#!/usr/bin/env python

import sys

def main():
  sys.stdout.write('{"changed":true}')

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

Most basic playbook

```
- hosts: localhost
  connection: local
  tasks:
  - most_basic_module:
```



```
(venv) ➜  ansible-custom-modules-demo ansible-playbook most_basic_playbook.yml

PLAY [localhost] ***************************************************************

TASK [setup] *******************************************************************
ok: [localhost]

TASK [most_basic_module] *******************************************************
changed: [localhost]

PLAY RECAP *********************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0
```

AnsibleModule

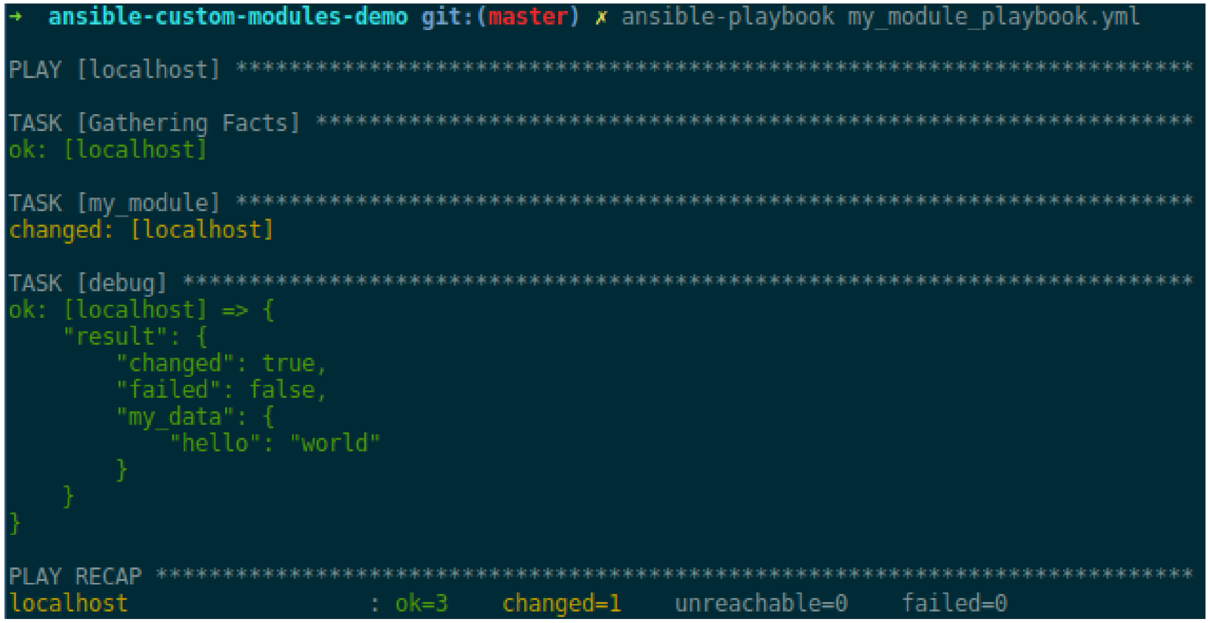
```
# library/my_module.py

#!/usr/bin/env python

from ansible.module_utils.basic import AnsibleModule

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

  result = dict()
  result["changed"] = True

  module.exit_json(**result)

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

Custom output (1)

```
# library/my_module.py

#!/usr/bin/env python

from ansible.module_utils.basic import AnsibleModule

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

  result = dict()
  result["changed"] = True
  result["my_data"] = {"hello": "world"}

  module.exit_json(**result)

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

Custom output (2)

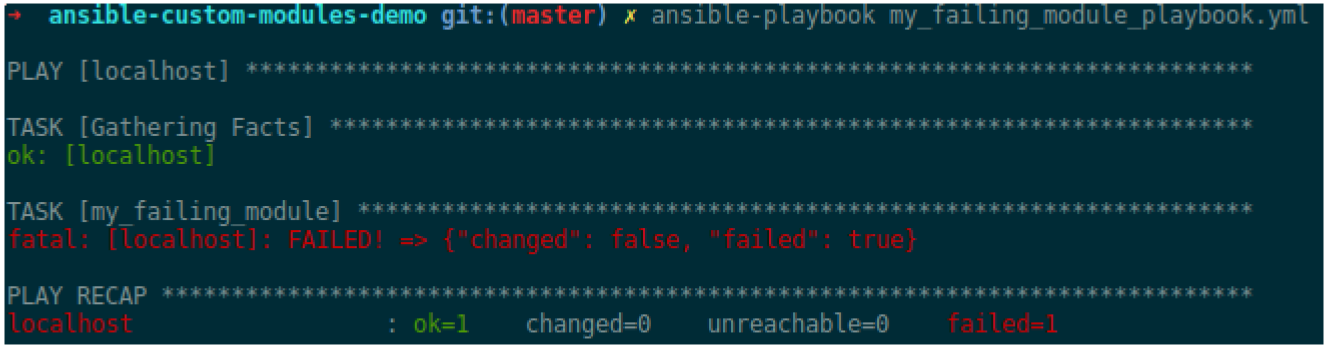
```
- hosts: localhost
  connection: local
  tasks:
  - my_module:
    register: result
  - debug: var=result
```



```
→  ansible-custom-modules-demo git:(master) ✗ ansible-playbook my_module_playbook.yml

PLAY [localhost] ***********************************************************************

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

TASK [my_module] ***********************************************************************
changed: [localhost]

TASK [debug] ***************************************************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false,
        "my_data": {
            "hello": "world"
        }
    }
}

PLAY RECAP *****************************************************************************
localhost                  : ok=3    changed=1    unreachable=0    failed=0
```

Failing

```
# library/my_failing_module.py

#!/usr/bin/env python

from ansible.module_utils.basic import AnsibleModule

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

  result = dict()
  result["failed"] = True

  module.exit_json(**result)

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



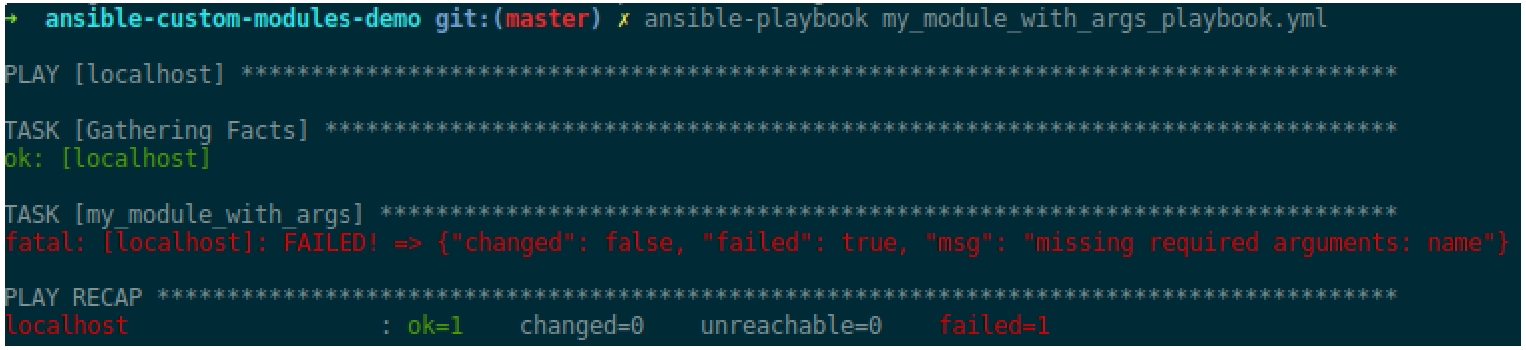
```
➜  ansible-custom-modules-demo git:(master) ✗ ansible-playbook my_failing_module_playbook.yml

PLAY [localhost] ***********************************************************************

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

TASK [my_failing_module] ***************************************************************
fatal: [localhost]: FAILED! => {"changed": false, "failed": true}

PLAY RECAP *****************************************************************************
localhost                  : ok=1    changed=0    unreachable=0    failed=1
```

Arguments (1)

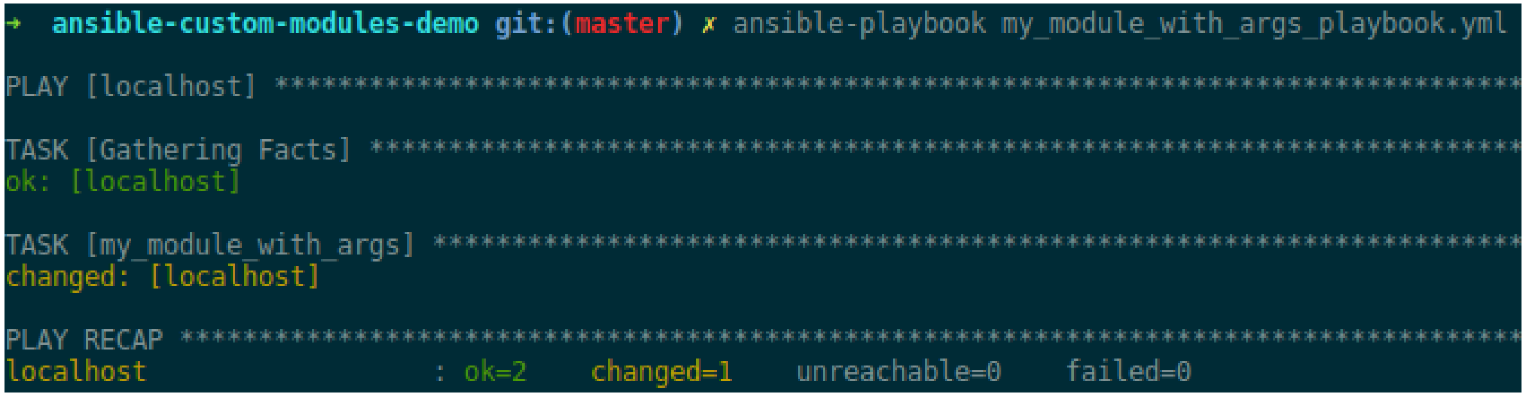
```python
# library/my_module_with_args.py

# ...
argument_spec = {
  'name': {'type': 'str', 'required': True},
  'scores': {'type': 'list', 'required': False, 'default': []}
}

module = AnsibleModule(
  argument_spec=argument_spec,
  supports_check_mode=False
)

name = module.params["name"]
scores = module.params["scores"]
#...
```

Arguments (2)

```
- hosts: localhost
  connection: local
  tasks:
  - my_module_with_args: name="florin" scores="{{ [1, 2, 3] }}"
```



```
→ ansible-custom-modules-demo git:(master) ✗ ansible-playbook my_module_with_args_playbook.yml

PLAY [localhost] ****************************************************************************

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

TASK [my_module_with_args] ******************************************************************
fatal: [localhost]: FAILED! => {"changed": false, "failed": true, "msg": "missing required arguments: name"}

PLAY RECAP **********************************************************************************
localhost                  : ok=1    changed=0    unreachable=0    failed=1
```



```
→ ansible-custom-modules-demo git:(master) ✗ ansible-playbook my_module_with_args_playbook.yml

PLAY [localhost] ****************************************************************************

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

TASK [my_module_with_args] ******************************************************************
changed: [localhost]

PLAY RECAP **********************************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0
```

Arguments (3)

```
# library/my_module_with_args.py

# ...
argument_spec = {
  'name': {'type': 'str', 'required': True},
  'scores': {'type': 'list', 'required': False, 'default': []},
  'state': {'choices': ['present', 'absent'], 'default': 'present'},
  'options': {'type': 'dict', 'default': {}},
}
#...
```

Arguments (4)

```
# library/my_module_with_more_args.py

# ...
  module = AnsibleModule(
    argument_spec=argument_spec,
    required_together=[
      ['name', 'scores'],
    ],
    required_one_of=[
      ['state', 'options']
    ],
    mutually_exclusive=[
      ['name', 'options']
    ],
    supports_check_mode=False
  )
#...
```

Other languages: Ruby (1)

```ruby
# library/my_ruby_module.rb

#!/usr/bin/env ruby

require "json"

params = {}

arguments = File.read(ARGV[0])
arguments.split(" ").each do |argument|
  key, value = argument.split("=")

  next unless key && value

  params[key] = value
end

# This is how you fetch parameters
params["name"]

puts ({ changed: true, params: params }).to_json
```

Other languages: Ruby (2)

```
- hosts: localhost
  connection: local
  tasks:
  - my_ruby_module: name="florin" state="present"
    register: result
  - debug: var=result
```

```
<1.2.3.4> PUT /tmp/tmpsZsEbg TO /home/ubuntu/.ansible/tmp/ansible-tmp-
1511273742.3-225952978315588/my_ruby_module.rb
```

Recap

- Pass, cast and validate arguments
- (Use different programming languages)
- Hide complexity
- Build "user"-friendly interfaces

A "user"-friendly interface?

```
# our base image
FROM alpine:3.5

# Install python and pip
RUN apk add --update py2-pip

# upgrade pip
RUN pip install --upgrade pip

# install Python modules needed by the Python app
COPY requirements.txt /usr/src/app/
RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt

# copy files required for the app to run
COPY app.py /usr/src/app/
COPY templates/index.html /usr/src/app/templates/

# tell the port number the container should expose
EXPOSE 5000

# run the application
CMD ["python", "/usr/src/app/app.py"]
```

Something simple…

Building infrastructure

```
- hosts: localhost
  connection: local
  tasks:
  - microservice:
      name: "some-service"
      instance_type: "t2.small"
      instance_count: 2
      db_instance_type: "db.t2.small"
      db_engine: "postgres"
      db_storage: 5
      public_lb: yes
      state: "present"
```

How?

- Use (sane) defaults
- Use conventions (e.g. DNS)
- Hide complexity
- Idempotence

Questions?

https://github.com/lipanski/ansible-custom-modules-demo

Reference

- http://docs.ansible.com/ansible/latest/list_of_all_modules.html
- http://docs.ansible.com/ansible/latest/dev_guide/developing_modules.html
- http://docs.ansible.com/ansible/latest/dev_guide/developing_modules_general.html